

Figure 1

FIG. 2

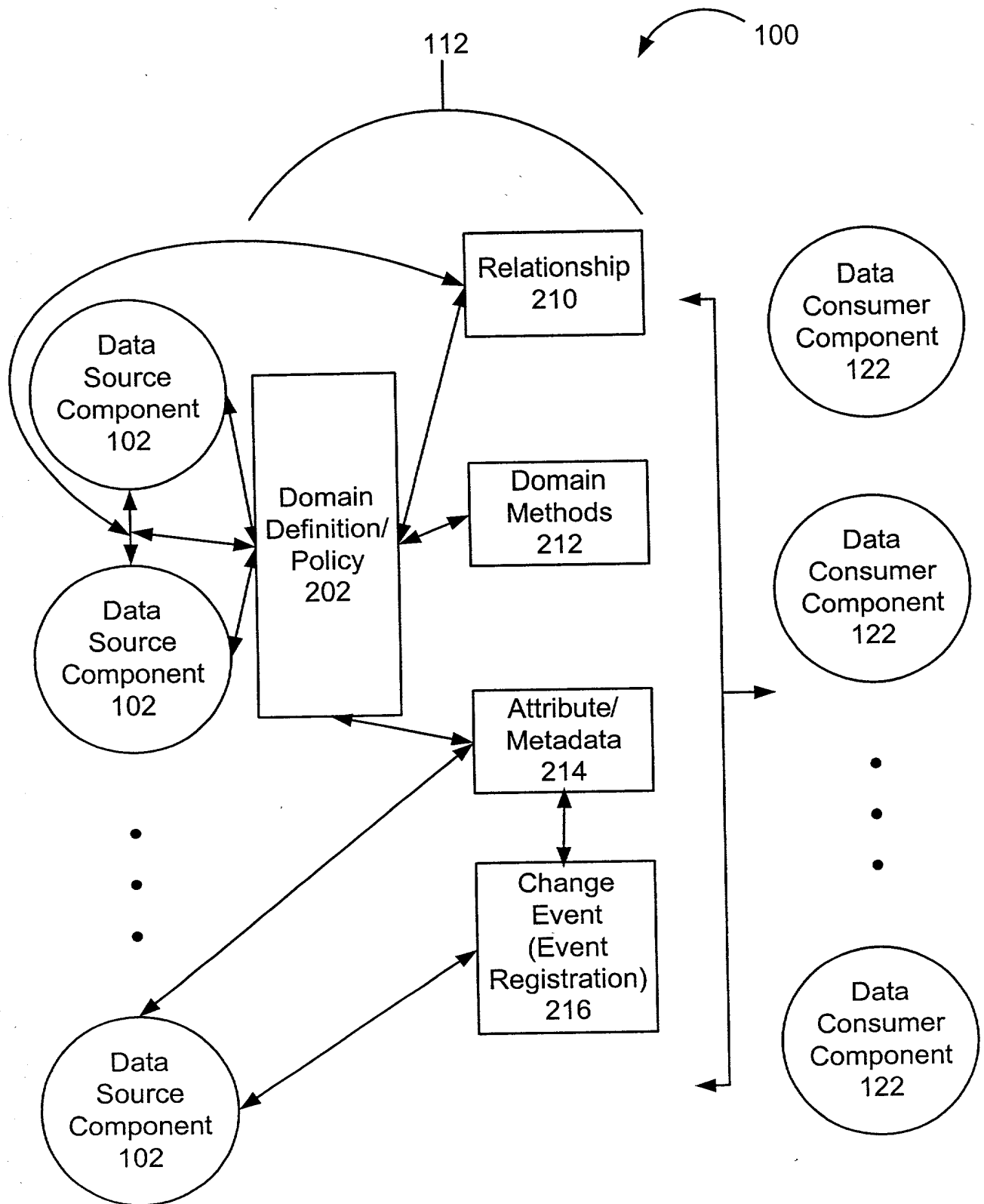
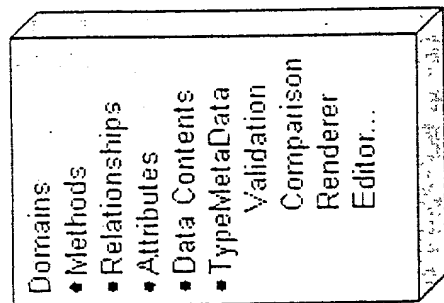


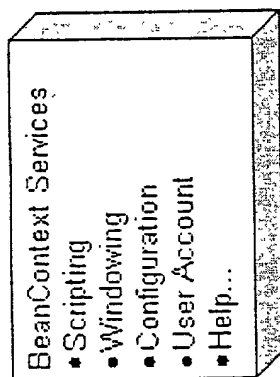
Figure 2

FIGURE 3A

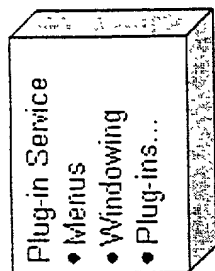
358



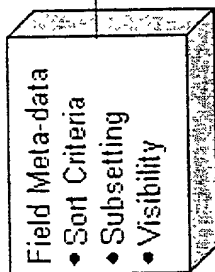
366



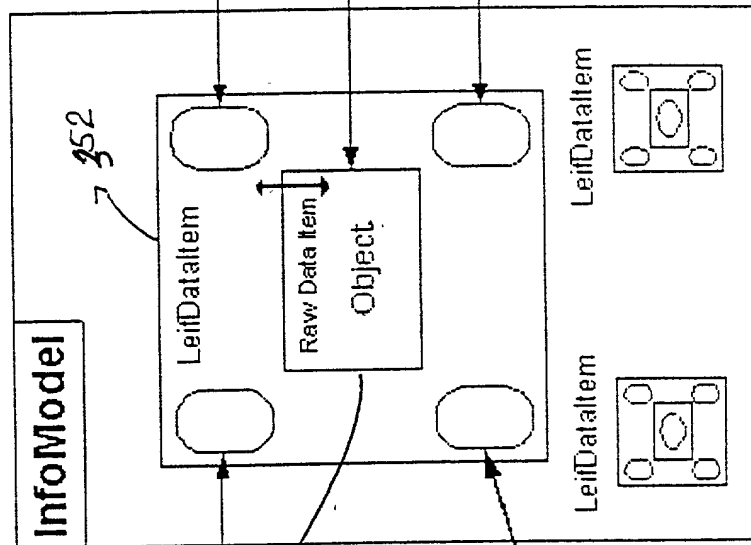
364



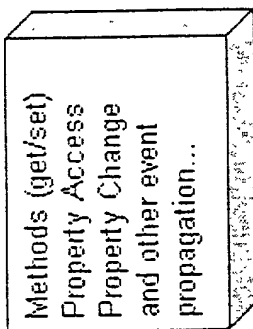
354



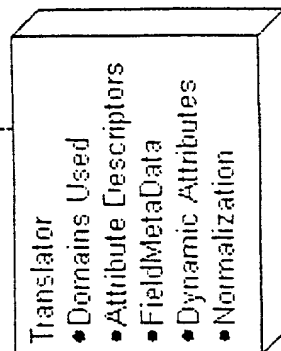
350



360



362



351

356

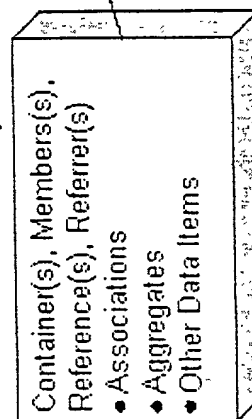
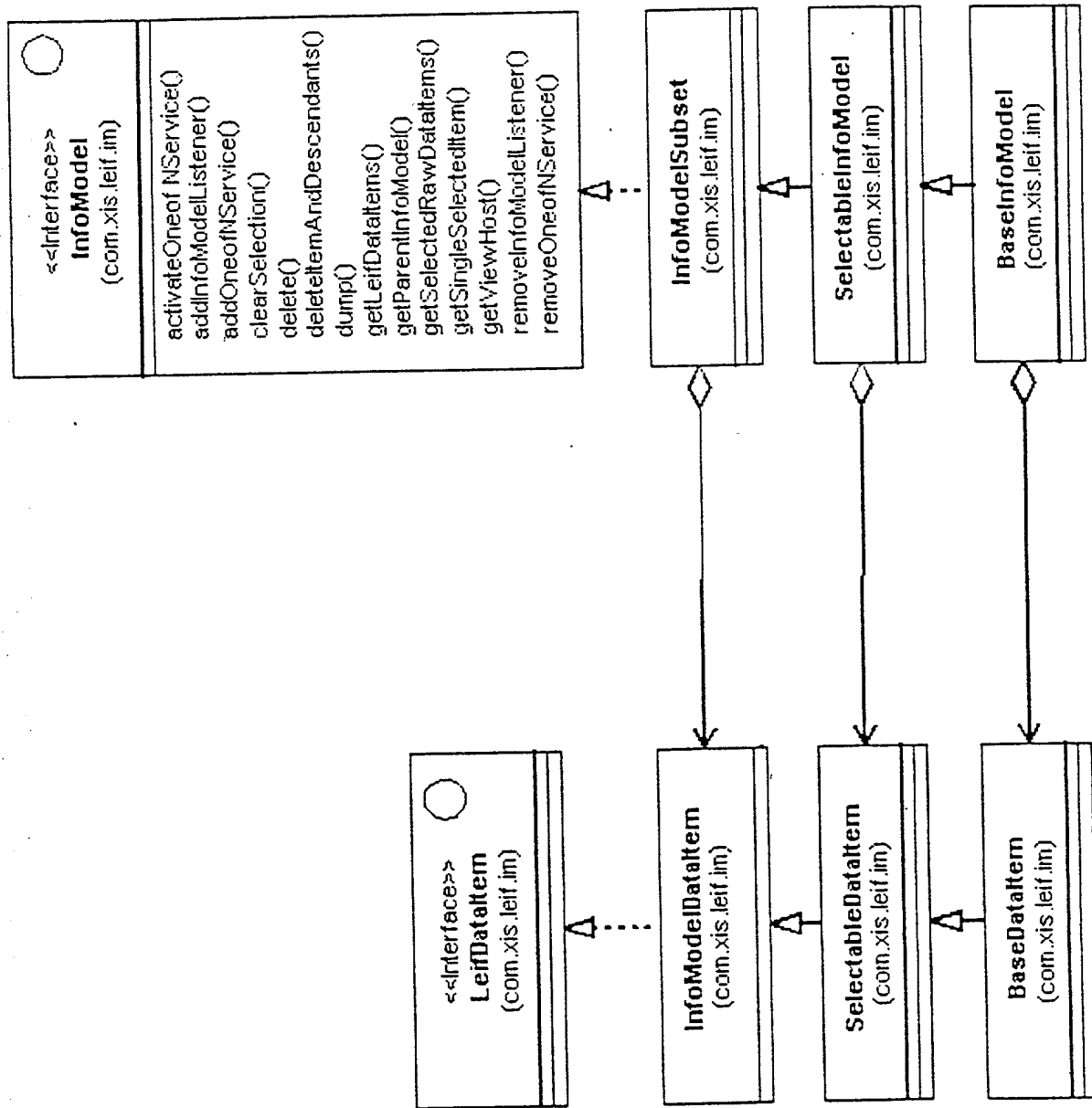


FIGURE 3B



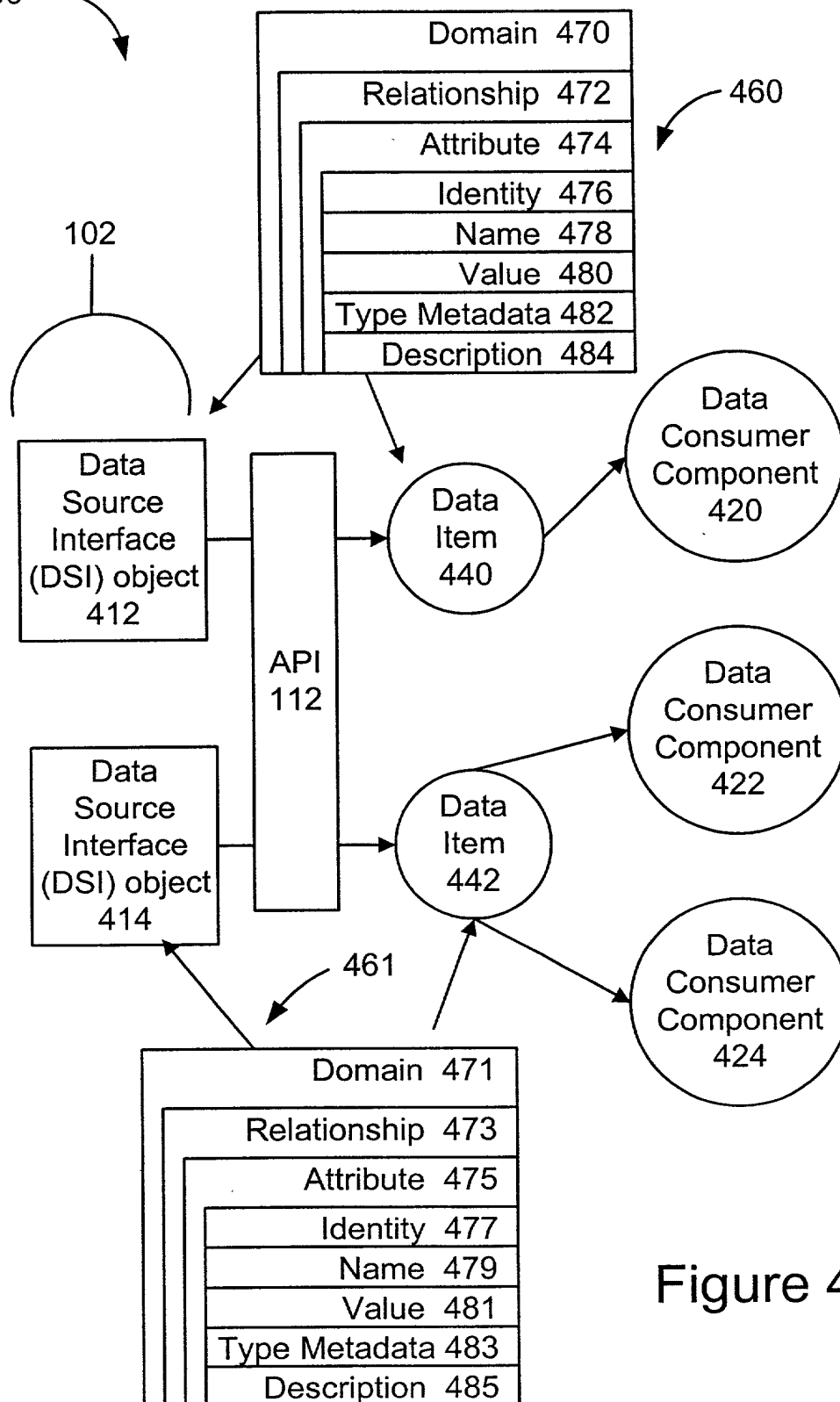


Figure 5

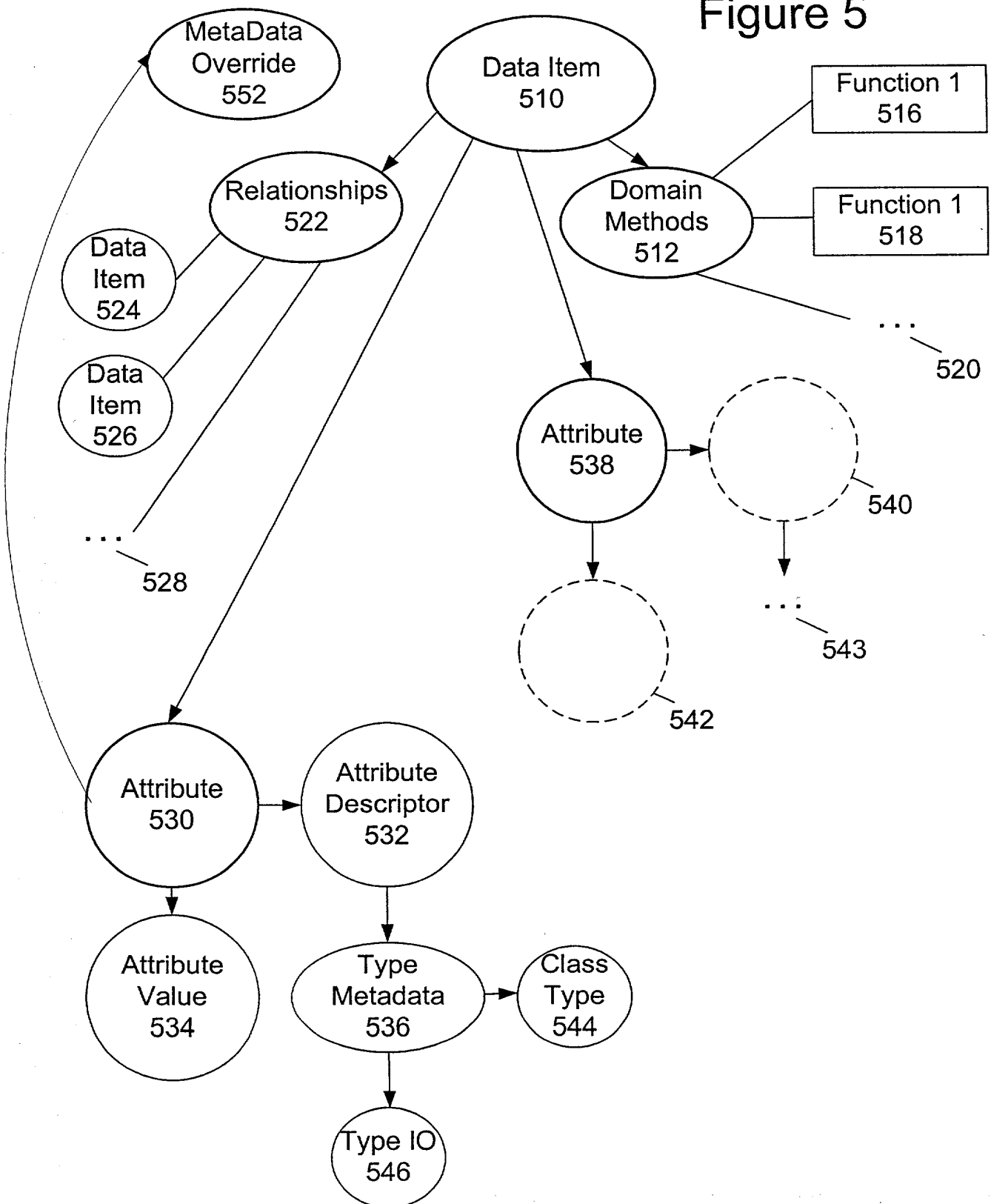
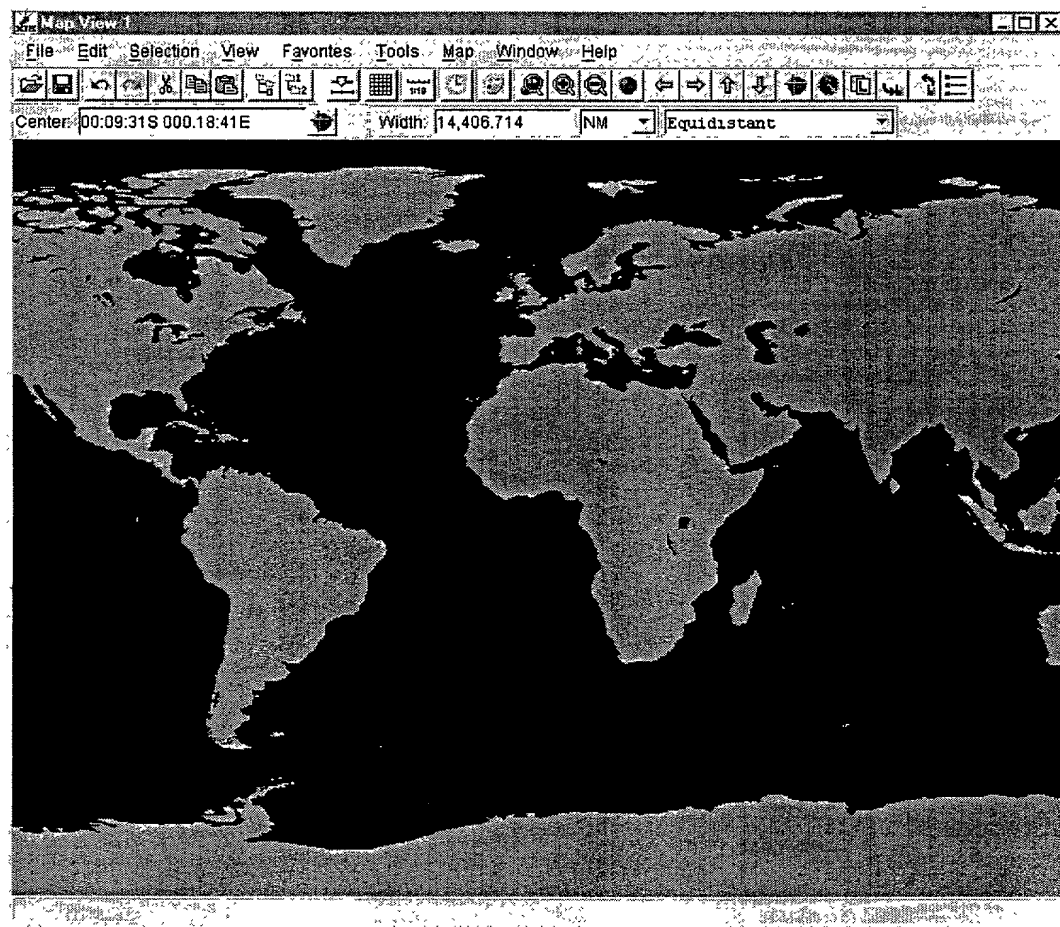


Figure 6A



10039300 102201 90E66001

Figure 6B

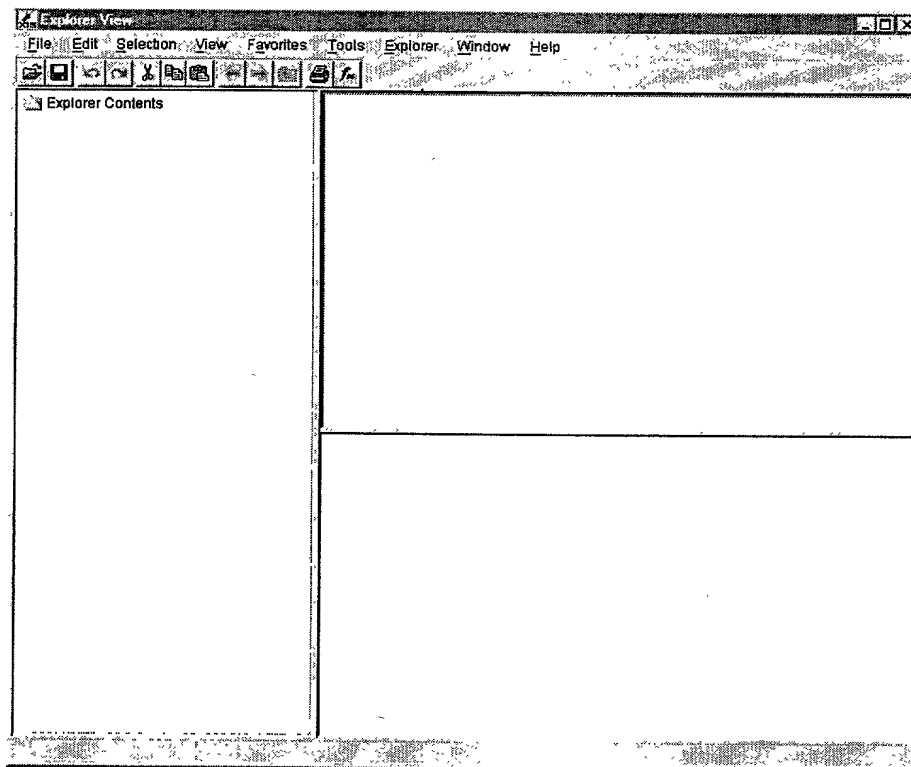




Figure 6C

People Source - XIS Explorer

File Edit Selection View Favorites Tools Explorer Window Help

Explorer Contents

- Orders
- Computer
- People Source

Name	Address	City	State	Zip Code	Te
Susan Sara...	85 Elm Street	Stratford	IL	23935	(899)4
Tim Curry	2416 Camino Del Mar	San Diego	CA	92381	(858)6
Barry Bost...	122 Thames Street	Newport	RI	14285	(401)6
Richard O...	33 Lockwood Avenue	Preston	PA	15003	(239)6
Patricia Quinn	1288 Huntington Avenue	Boston	MA	02119	(617)2
Nell Campbell	444 Mix Avenue	Hamden	CT	03419	(203)6
Jonathan ...	1 Quincy Place	Quincy	MA	02086	(617)6
Peter Hin...	236 Milsure Street	Sedona	AZ	84921	(651)6
Meat Loaf	13 Gwendal Avenue	Los Angeles	CA	92114	(818)6
Charles Gray	3929 General Street	Littleton	NH	03561	(603)4
Jeremy N...	118 Cornwall Place	Stallion	TX	43208	(838)2
Hilary Labow	9 Overlook Drive	Medway	MA	02053	(508)6
Bruce Rnx	44 Ford Lane	Terminon	WA	84937	(829)6

Preferred Attributes

Property	Value
Name	People Source
Children Are Same Type	<input type="checkbox"/>
Susan Sarandon	
Name	Susan Sarandon
Address	85 Elm Street
City	Stratford
State	IL
Zip Code	23935

Figure 6D

Order #70 - XIS Explorer

File Edit Selection View Favorites Tools Explorer Window Help

Orders

- Order #67
- Order #70
- Order #72
- Order #74
- Order #76
- Order #83
- Order #87
- Order #94
- Order #96
- Order #105
- Order #109
- Order #117
- Order #118
- Order #122
- Order #128
- Order #132
- Order #141
- Order #143
- Order #145
- Order #155
- Order #159
- Order #168
- Order #170
- Order #173
- Order #177
- Order #190
- Order #200
- Order #204
- Order #205
- Order #213
- Order #214
- Order #218
- Order #220
- Order #223

Name	Quantity	Retail Price (USD)	Product ID
Basketball (#1)	7	\$4.95	1
Soccer ball (#3)	12	\$12.95	3
Tether ball (#9)	14	\$9.65	9

Preferred Attributes

Property	Value
Name	Order #70
Number of Items	33
Total (USD)	\$27.55
Payment Method	Check
Processing Time (DAY)	3
Location	42:27:58N 083:11:02W
Pen Color	<input type="checkbox"/>
Zipcode	48237

Apply Reset

Figure 6E

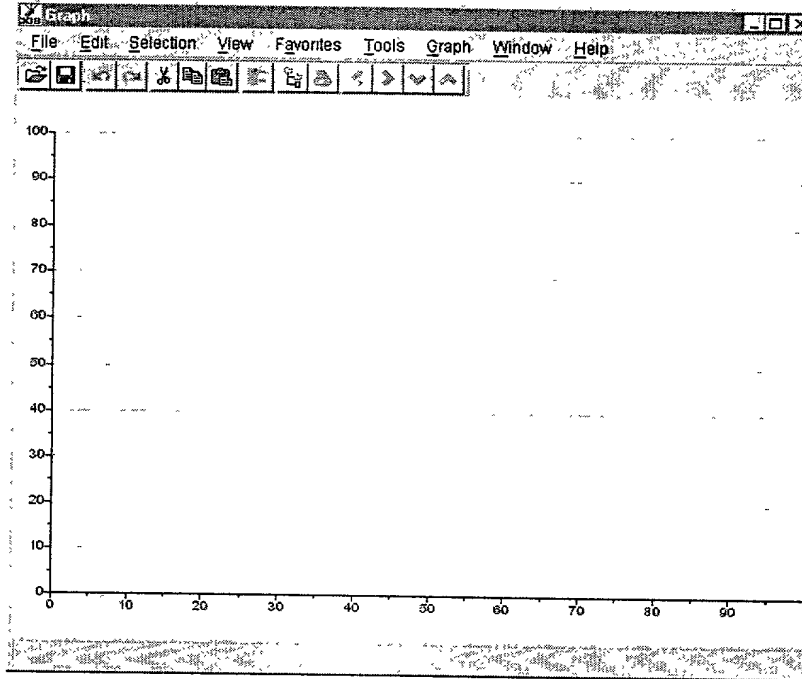
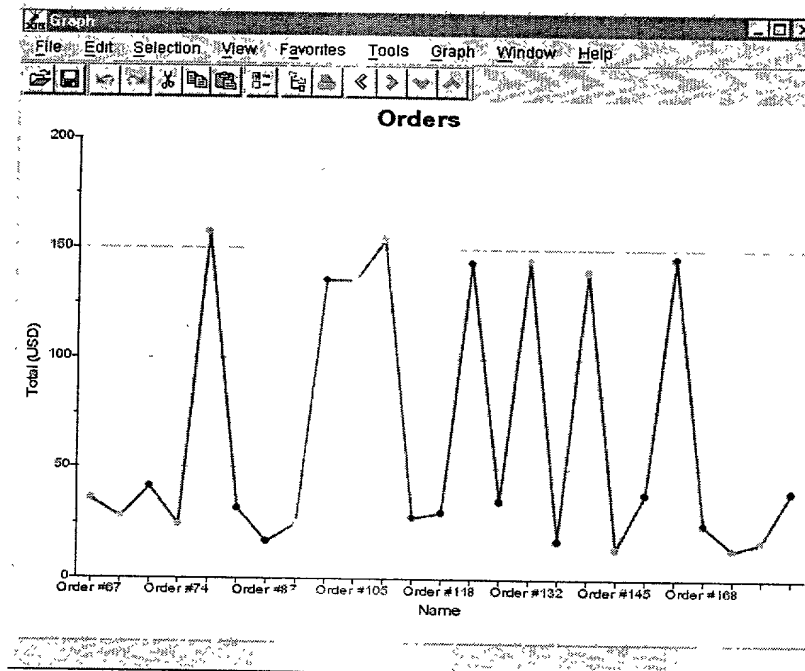


Figure 6F



# Figure 7

The screenshot shows a window titled "Property Sheet - Commander". Inside, there is a section labeled "Preferred Properties" with a small icon to its right. Below this is a table with two columns: "Property" and "Value".

Property	Value
SSN	555-55-5555
Display Name	Commander
DOB	Sep 4, 2010 5:00:02 AM
Name	Commander

At the bottom of the dialog are four buttons: "Ok", "Cancel", "Reset", and "Apply".

# Figure 8

```

Person
(com.XIS.test)

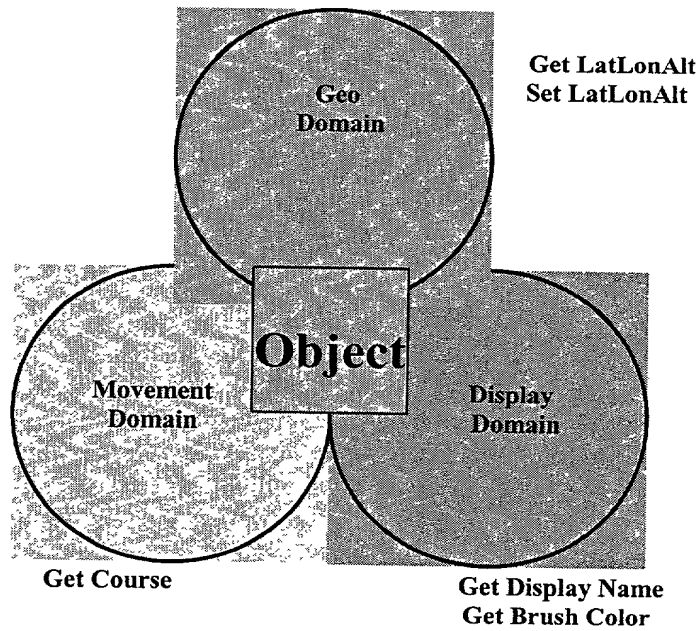
-dob : Date
-name : String
-ssn : String

+getDOB() : Date
+getName() : String
+getSSN() : String
+Person(name : String)
+Person()
+setDOB(dtg : Date) : void
+setName(newName : String) : void
+setSSN(newSSN : String) : void
+toString() : String
    
```

# Figure 9

getAttributes()	References	Referrers	Members
com.xis.domains.display.DisplayDomain.displayName	None	None	None
com.xis.test.Person.DOB			
com.xis.test.Person.SSN			
com.xis.test.Person.name			

# Figure 10



10039306-102201

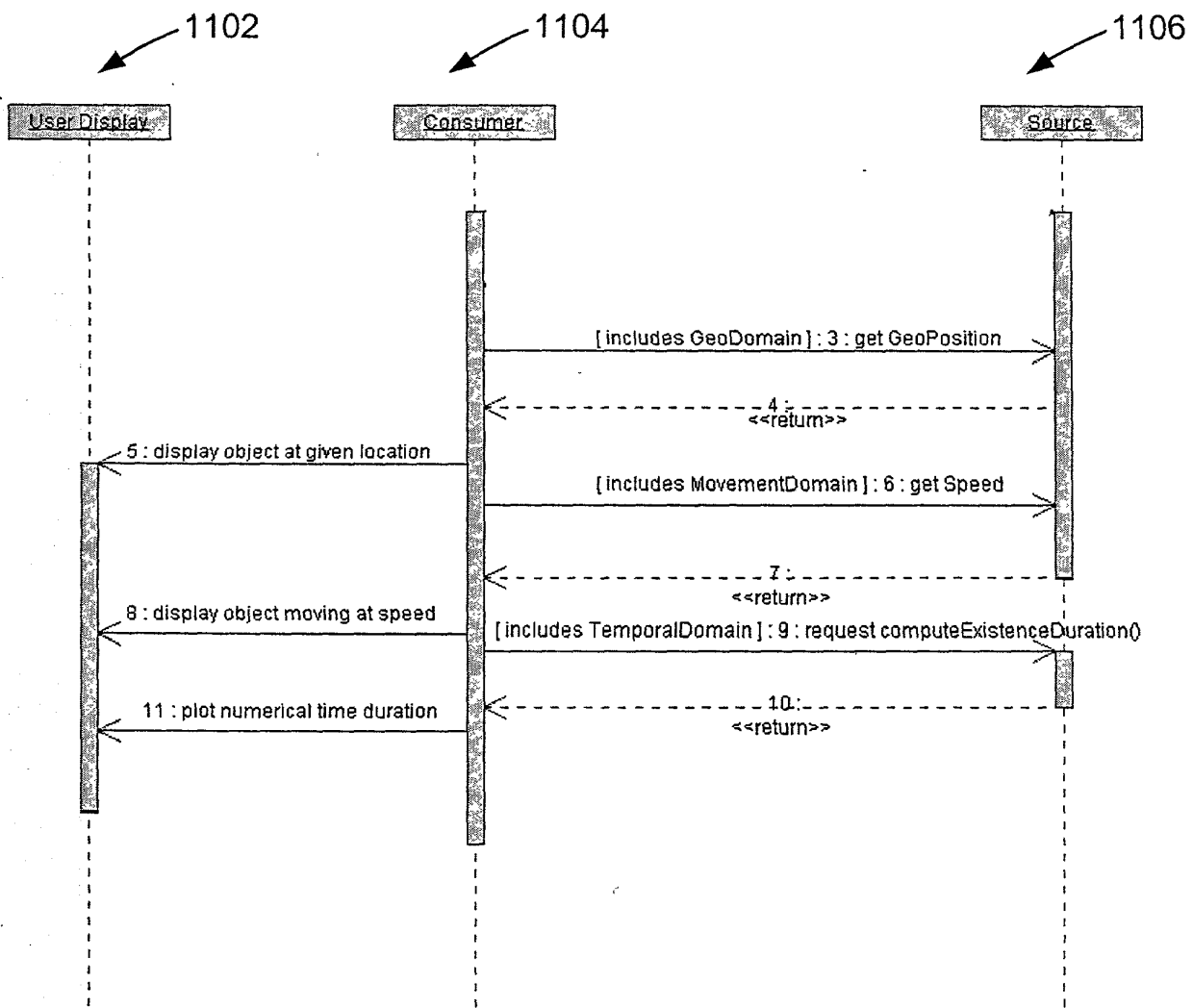


Figure 11

## Package com.xis.types

This package contains classes that provide several standard TypeMetaData classes for describing types and their constraints, and for rendering and editing values of those types.

See:

Description

Interface Summary	
<u>DataTest</u>	The DataTest interface specifies methods for Object validation.
<u>HTMLTypeIO</u>	This interface defines the IO for HTML.
<u>SummaryFunction</u>	The SummaryFunction interface defines generic summary functionality based upon provided input data values.
<u>SwingTypeIO</u>	The SwingTypeIO interface allows for the use of both swing editors, allowing swing components to edit an object, and swing renderers, which know how to render these objects in a swing environment.
<u>TextTypeIO</u>	The TextTypeIO interface provides a means of formatting objects in a textual fashion, as well as parsing text from which an object is created.
<u>TypedValue</u>	The TypedValue interface is used to hold an object that carries its own TypeMetaData with it.
<u>TypeEditor</u>	The TypeEditor interface defines methods for editing attributes provided by the types implemented within this package.
<u>TypeIO</u>	The TypeIO interface provides a common base from which other TypeIOs can extend, such as HTMLTypeIO, SwingTypeIO, etc.
<u>TypeMetaData</u>	The TypeMetaData interface defines generic type accessors for object comparing, editing, formatting, rendering, and validation.
<u>TypeMetaDataFactory</u>	The TypeMetaDataFactory interface defines a class that can create TypeMetaData for a given Class Type.
<u>TypeRenderer</u>	The TypeRenderer interface defines methods for rendering attributes provided by the types implemented within this package.
<u>ValidTestProxy</u>	The ValidTestProxy interface
<u>WMLTypeEditor</u>	The WMLTypeEditor interface defines methods for rendering attributes provided by the types implemented within this package.
<u>WMLTypeIO</u>	This interface defines the IO for WML format.
<u>WMLTypeRenderer</u>	The WMLTypeRenderer interface defines methods for rendering attributes provided by the types implemented within this package.
<u>XMLTypeIO</u>	The XMLTypeIO interface provides a means of formatting objects in a XML textual fashion, as well as parsing XML text for creating an object.

FIG. 12A

## Class Summary

<u>AbstractDataTest</u>	The AbstractDataTest class provides a default implementation of <i>test (Object)</i>
<u>AbstractTypeMetaData</u>	AbstractTypeMetaData provides a partial implementation of TypeMetaData to relieve the XIS developer from explicitly implementing irrelevant methods.
<u>AreaOfUncertaintyTypeMetaData</u>	AreaOfUncertaintyTypeMetaData is a type object that supports <i>AreaOfUncertainty</i> objects.
<u>ArrayListTypeMetaData</u>	ArrayListTypeMetaData is a type object that supports <i>java.util.ArrayList</i> objects.
<u>ArrayTypeMetaData</u>	ArrayTypeMetaData is a type object that supports <i>java.lang.reflect.Array</i> objects.
<u>BeanTypeMetaData</u>	BeanTypeMetaData is a type object that supports <i>java.beans</i> objects.
<u>BooleanTypeMetaData</u>	The BooleanTypeMetaData is a type object that supports <i>Boolean</i> objects.
<u>BooleanTypeMetaDataFactory</u>	A BooleanTypeMetaDataFactory can create a BooleanTypeMetaData for given booleans.
<u>CachedTypeMetaData</u>	CachedTypeMetaData is a type object that simply delegates all TypeMetaData calls to another TypeMetaData.
<u>ClassificationTypeMetaData</u>	ClassificationTypeMetaData is a type object that supports <i>Classification</i> objects.
<u>CollectionsTypeMetaData</u>	CollectionsTypeMetaData is a type object that supports <i>Collection</i> objects.
<u>ColorTypeMetaData</u>	The ColorTypeMetaData class implements TypeMetaData for <i>Color</i> objects.
<u>ColorTypeMetaDataFactory</u>	A ColorTypeMetaDataFactory can create a ColorTypeMetaData for a given <i>Color</i> object.
<u>ConversionNumericTypeMetaData</u>	A generic NumericTypeMetaData for converting from one unit to another.
<u>CurrencyTypeMetaData</u>	CurrencyTypeMetaData is a type object that supports <i>Number</i> objects that represent <i>Currency</i> values.
<u>DateTimeTypeMetaData</u>	DateTimeTypeMetaData is a type object that supports <i>Date</i> objects.
<u>DateTimeTypeMetaDataFactory</u>	A DateTimeTypeMetaDataFactory can create a DateTimeTypeMetaData for given <i>Date</i> objects.
<u>DiscreteRangeStringTypeMetaData</u>	DiscreteRangeStringTypeMetaData is a type object that supports <i>String</i> objects with discrete ranges.
<u>DisplayLabelTypeMetaData</u>	DisplayLabelTypeMetaData is a type object that supports supports <i>DisplayLabel</i> objects.
<u>DTGTypeMetaData</u>	DTGTypeMetaData is a type object that supports <i>Date</i> objects.
<u>EnumerationType</u>	Class to implement an enumeration in Java.
<u>EnumerationTypeMetaData</u>	The EnumerationTypeMetaData class is used to represent integer constants as strings to the user.
<u>FontTypeMetaData</u>	FontTypeMetaData is a type object that supports <i>Font</i> objects.
<u>HashMapTypeMetaData</u>	HashMapTypeMetaData is a type object that supports <i>java.util.HashMap</i> objects.

FIG. 12 B

10039306 " 102201



<u>HashSetTypeMetaData</u>	HashSetTypeMetaData is a type object that supports <i>java.util.HashSet</i> objects.
<u>IconShapeTypeMetaData</u>	IconShapeTypeMetaData is a type object that supports <i>IconShape</i> objects.
<u>IconTypeMetaData</u>	IconTypeMetaData is a type object that supports <i>Icon</i> objects.
<u>LinkedListTypeMetaData</u>	LinkedListTypeMetaData is a type object that supports <i>java.util.LinkedList</i> objects.
<u>ListTypeMetaData</u>	ListTypeMetaData is a type object that supports <i>java.util.List</i> objects.
<u>MouseMapProxy</u>	The MouseMapProxy class allows TypeEditors access to the map without requiring them to having any compile time knowledge of the map's existence.
<u>NumberComparator</u>	An implementation of the Comparator interface that compares two objects that extend Number, or that both implement Comparable, with the class of one assignable from the other.
<u>NumericTypeMetaData</u>	NumericTypeMetaData is a type object that supports <i>Number</i> objects
<u>NumericTypeMetaDataFactory</u>	A NumericTypeMetaDataFactory can create NumericTypeMetaData for a given class type or method return type.
<u>ObjectTypeMetaData</u>	ObjectTypeMetaData is a type that supports a simple <i>Object</i> and is provided to quickly add arbitrary attribute types to a Data Source Interface without writing a more specific type handler.
<u>PercentTypeMetaData</u>	PercentTypeMetaData is a type object that supports <i>Number</i> objects that represent Percent(%) values.
<u>ProbabilityTypeMetaData</u>	ProbabilityTypeMetaData is a type object that supports <i>Number</i> objects that represent Probability values.
<u>RenamedTypeMetaData</u>	RenamedTypeMetaData simply delegates all TypeMetaData calls to another TypeMetaData except for the <i>getName()</i> , which is overridden with the given value.
<u>ResizedTextTypeMetaData</u>	ResizedTextTypeMetaData simply delegates all TypeMetaData calls to another TypeMetaData except for the <i>getPixelWidth()</i> , which is overridden with the given value.
<u>Resources</u>	The Resources class is automatically generated and must be public, but it is intended to be used only by Java's internationalization support classes.
<u>SmartDurationTypeMetaData</u>	A SmartDurationTypeMetaData for converting from one time unit to another based on the magnitude of the duration value.
<u>StringTypeMetaData</u>	StringTypeMetaData is a type object that supports <i>String</i> objects.
<u>StringTypeMetaDataFactory</u>	A StringTypeMetaDataFactory can create a StringTypeMetaData for a given String object.
<u>TextTypeMetaData</u>	TextTypeMetaData is a type object that supports <i>Text</i> objects.
<u>TypedValueTypeMetaData</u>	TypedValueTypeMetaData is a type object that supports supports <i>TypedValue</i> objects.
<u>TypeEditorBeanContextChildSupport</u>	The TypeEditorBeanContextChildSupport class handles most of the responsibilities of a TypeEditor and a BeanContextChild.
<u>TypeEditorSupport</u>	The TypeEditorSupport class handles most of the responsibilities of a TypeEditor.
<u>TypeIOPluggableService</u>	The TypeIOPluggableService class is responsible for loading TypeIOs.
<u>TypeIORegistry</u>	The TypeIORegistry class is a registry for different implementations of TypeIO classes.

FIG. 12C

<u>TypeIOSupport</u>	The TypeIOSupport provides support for TypeMetaData's getTipo methods.
<u>TypeMetaDataDelegator</u>	TypeMetaDataDelegator is a type object that simply delegates all TypeMetaData calls to another TypeMetaData.
<u>TypeMetaDataFactoryPluggableService</u>	The TypeMetaDataFactoryPluggableService class is responsible for loading TypeMetaDataFactories.
<u>TypeMetaDataFactoryStore</u>	This Class stores TypeMetaDataFactories.
<u>TypePreferences</u>	This class is used by TypeMetaData instances to pass information about preferred TypeMetaData objects.
<u>Types</u>	The Types class is a holder class for the RESOURCES global variable for resource properties of the com.xis.types package.
<u>URLTypeMetaData</u>	URLTypeMetaData is a type object that supports URL objects.

### Exception Summary

<u>NoSuchEnumerationException</u>	Class to implement an enumeration exception in Java.
<u>ParseFailedException</u>	A ParseFailedException is thrown (typically by Tipo objects) when it is not possible to parse a given String as desired.
<u>TestFailedException</u>	The TestFailedException is thrown from the "test()" method of a DataTest subclass when the value fails the test.

### Package com.xis.types Description

This package contains classes that provide several standard TypeMetaData classes for describing types and their constraints, and for rendering and editing values of those types.

FIG. 12D

FIG. 13

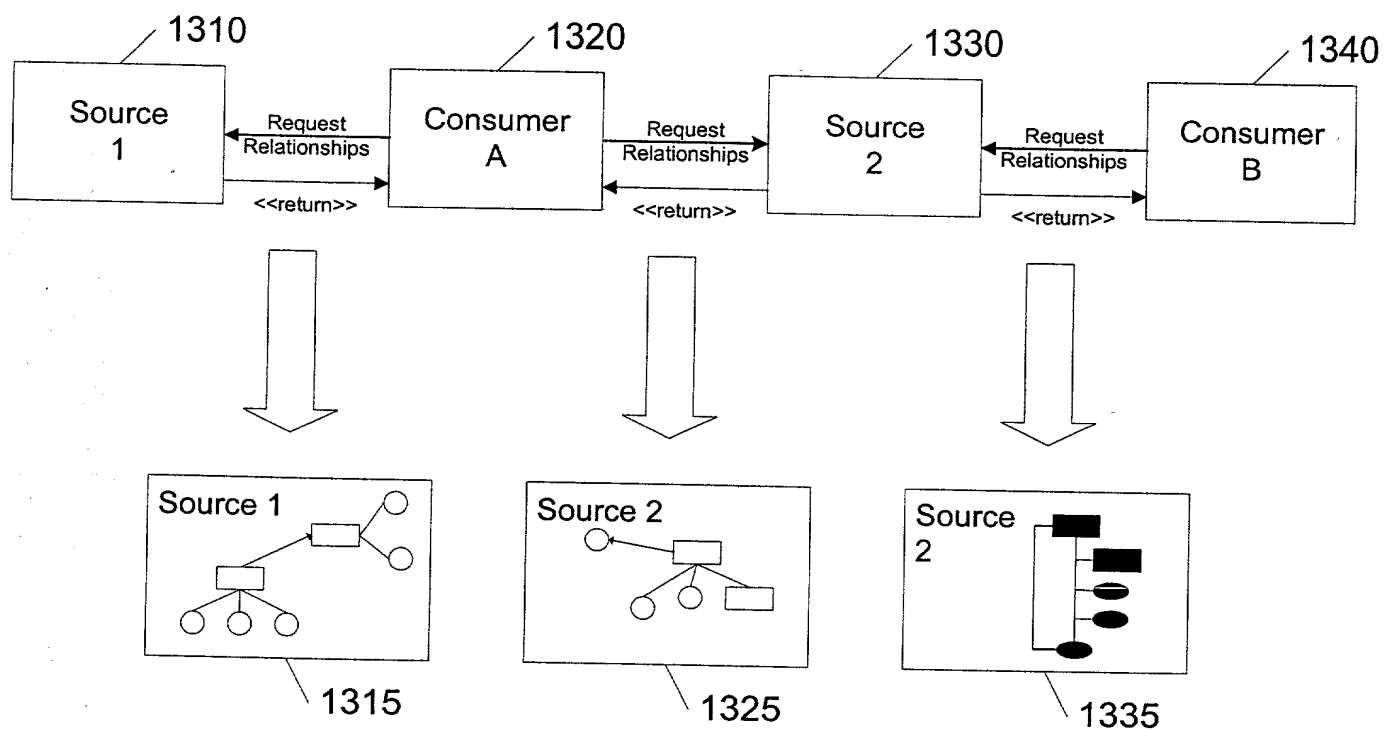
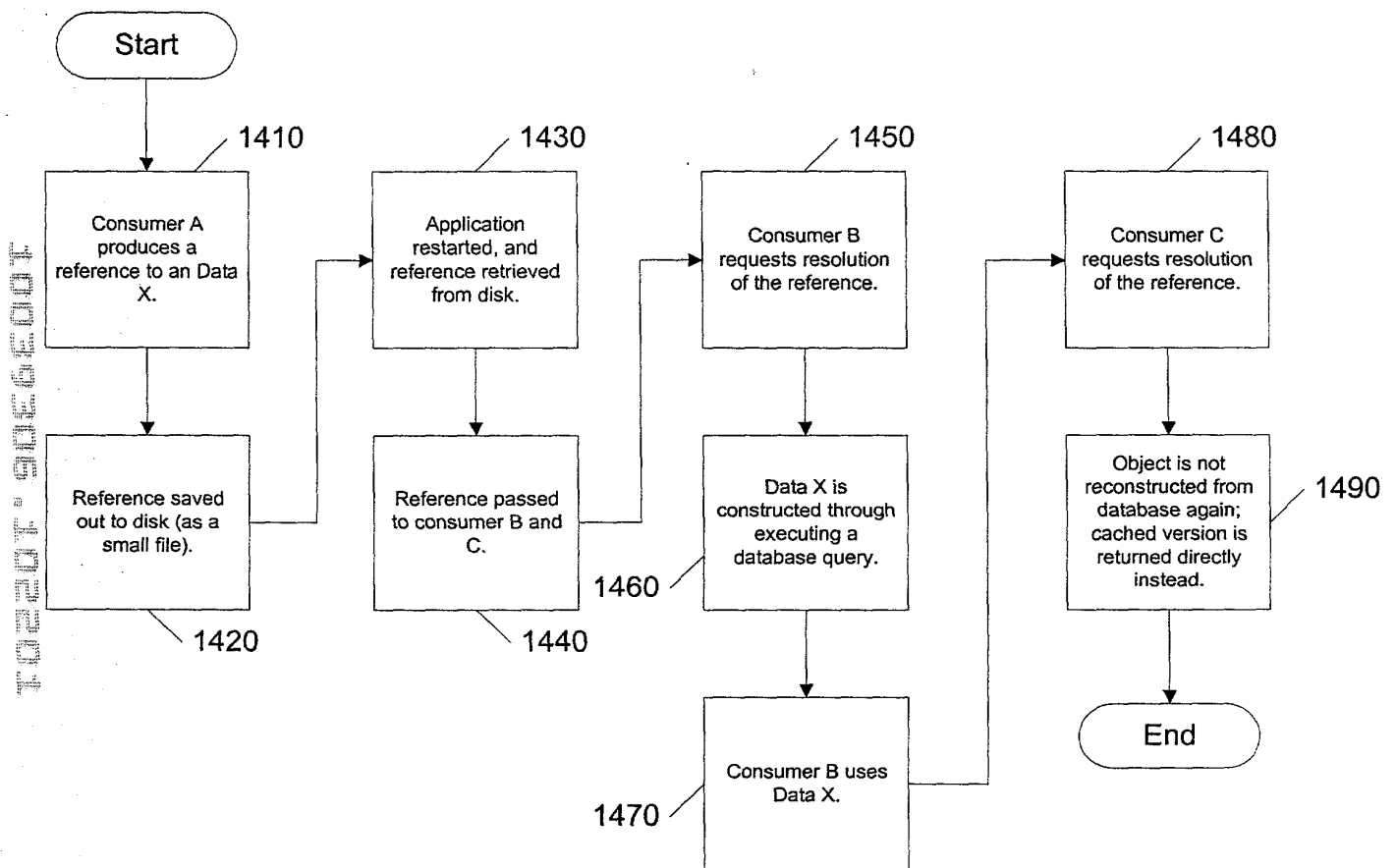


Figure 13



### Figure 14

# Figure 15

## HelloWorld.java

/\* XIS Tutorial standalone sequence example 1 data class. \*/

```
public class HelloWorld {  
    private float value = 1.5f;  
  
    public String toString() {  
        return "Hello World!";  
    }  
  
    public int getID() {  
        return 5;  
    }  
  
    public float getValue() {  
        return value;  
    }  
  
    // uncomment this to make "value" editable  
    /*  
    public void setValue(float value) {  
        this.value = value;  
    }  
    */  
}
```

## Figure 16A

### TestHarness.java

```
/* XIS Tutorial standalone sequence example 1 XIS interfacing. */
import javax.swing.JFrame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import com.xis.propertysheet.PropertySheetInfoBean;
import com.xis.ui.UIBeanEvent;
import com.xis.ui.UIBeanAdapter;
import com.xis.leif.im.BaseInfoModel;

public class TestHarness {

    public static void main(String[] args) {

        // the plugin manager is only required for more complex applications
        // involving multiple components integrated at runtime
        BaseInfoModel.setStartingPluginManager(false);

        // a property sheet infobean to display HelloWorld's attributes
        PropertySheetInfoBean properties = new PropertySheetInfoBean();
        properties.addRowDataItem(new HelloWorld());

        // add listener for 'OK,' 'cancel,' or close, which generate 'close' events
        properties.addUIBeanListener(
            new UIBeanAdapter() {
                public void closed(UIBeanEvent event) {
                    System.exit(0);
                }
            }
        );

        // a top-level frame to hold our property sheet infobean
        JFrame frame = new JFrame("HelloWorld Properties");
        // add a listener for window closing
        frame.addWindowListener(
            new WindowAdapter() {
                public void windowClosing(WindowEvent e) {
                    System.exit(0);
                }
            }
        );
    }
}
```

1602

1604

1606

1608

1610A

### Figure 16B

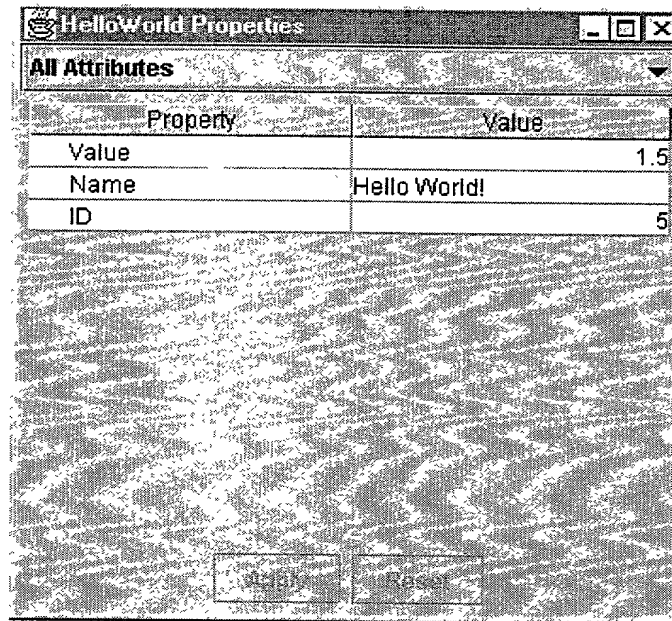
### Continuation of TestHarness.java

```
// stick the bean in the frame and display it
frame.getContentPane().add(properties);
frame.pack();
frame.setVisible(true);
```

1610B

[illegible]

Figure 17



The screenshot shows a window titled "HelloWorld Properties" with a standard Windows-style title bar (minimize, maximize, close buttons). Below the title bar is a dropdown menu currently set to "All Attributes". The main area contains a table with two columns: "Property" and "Value". The table has three rows of data. Below the table, there are two buttons: "Apply" and "Reset".

Property	Value
Value	1.5
Name	Hello World!
ID	5

1002900 90E6E00F



# Figure 18A

## HelloWorld.java

/\* XIS Tutorial standalone sequence example 2 data class. \*/

/\*{\*/

import java.awt.Color;  
import java.beans.PropertyChangeSupport;  
import java.beans.PropertyChangeListener;  
/\*}\*/

public class HelloWorld {

/\*{\*/

private int value = 1;  
private Color myColor = Color.green;

// this member class helps distribute property change events within XIS  
private PropertyChangeSupport propertyChangeSupport =  
 new PropertyChangeSupport(this);

// two aux methods to let other XIS objects pay attention to this one  
public void addPropertyChangeListener(PropertyChangeListener l) {  
 propertyChangeSupport.addPropertyChangeListener(l);  
}

public void removePropertyChangeListener(PropertyChangeListener l) {  
 propertyChangeSupport.removePropertyChangeListener(l);  
}

public String toString() {  
 return "A HelloWorld Object";  
}

public String getGreeting() {  
 return "Hello World!";  
}

/\*}\*/

public int getID() {  
 return 5;  
}

10039306-102001

1806

1802

## Figure 18B

### Continuation of of HelloWorld.java

```
public /*{*/ int /*}*/ getValue() {  
    return value;  
}
```

```
/*{*/
```

```
public void setValue(int value) {  
    // only update and fire property change if this is really a change  
    if (this.value != value) {  
        int oldValue = this.value;  
        this.value = value;  
        // fire property change event to notify other XIS objects  
        propertyChangeSupport.firePropertyChange("value", oldValue, value);  
    }  
}
```

```
public Color getMyColor() {  
    return myColor;  
}
```

```
public void setMyColor(Color myColor) {  
    // only update and fire property change if this is really a change  
    if (this.myColor != myColor) {  
        Color oldMyColor = this.myColor;  
        this.myColor = myColor;  
        // fire property change event to notify other XIS objects  
        propertyChangeSupport.firePropertyChange("myColor",  
                                                    oldMyColor, myColor);  
    }  
}
```

```
/*}*/
```

```
}
```

# Figure 19A

## TestHarness.java

/\* XIS Tutorial standalone sequence step 2 XIS interfacing. \*/

```
import javax.swing.JFrame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import com.xis.propertysheet.PropertySheetInfoBean;
import com.xis.ui.UIBeanEvent;
import com.xis.ui.UIBeanAdapter;
import com.xis.leif.im.BaseInfoModel;
/*{*/
import jclass.chart.JCChart;
import com.xis.plot.PlotInfoBean;
import com.xis.plot.chartviews.LeifChartView;
/*}*/
```

```
public class TestHarness {
```

```
    public static void main(String[] args) {
```

```
        // the plugin manager is only required for more complex applications
        // involving multiple components integrated at runtime
        BaseInfoModel.setStartingPluginManager(false);
```

```
        HelloWorld hello = new HelloWorld();
```

```
        // a property sheet infobean to display HelloWorld's attributes
        PropertySheetInfoBean properties = new PropertySheetInfoBean();
        properties.addRowDataItem(hello);
```

```
        // add a listener for 'OK' or 'cancel', which generate 'close' events
```

```
        properties.addUIBeanListener(
            new UIBeanAdapter() {
                public void closed(UIBeanEvent event) {
                    System.exit(0);
                }
            }
        );
```

```
    }
```

1902

## Figure 19B

### Continuation of TestHarness.java

```
/*{*/
// a top-level frame to hold our property sheet infobean
JFrame propertySheetFrame = new JFrame("HelloWorld Properties");
// add a listener for window closing
propertySheetFrame.addWindowListener(
    new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    }
);

// stick the property Sheet bean in the frame and display it
propertySheetFrame.getContentPane().add(properties);
propertySheetFrame.pack();
propertySheetFrame.setVisible(true);

// now we create a plot infobean to plot HelloWorld's numeric attribute
PlotInfoBean plot = new PlotInfoBean();
plot.addRowDataItems(new Object[] { hello });
plot.setChartType(JCChart.BAR);
// the alternatives are SCATTER_PLOT, PLOT, AREA, PIE, CANDLE,
// and STACKING_BAR, though not all will make sense in this example

// a top-level frame as before to hold our property sheet infobean
JFrame plotFrame = new JFrame("HelloWorld Plot");

// stick the plot bean in and put it up
plotFrame.getContentPane().add(plot);
plotFrame.pack();
plotFrame.setVisible(true);
/*}*/

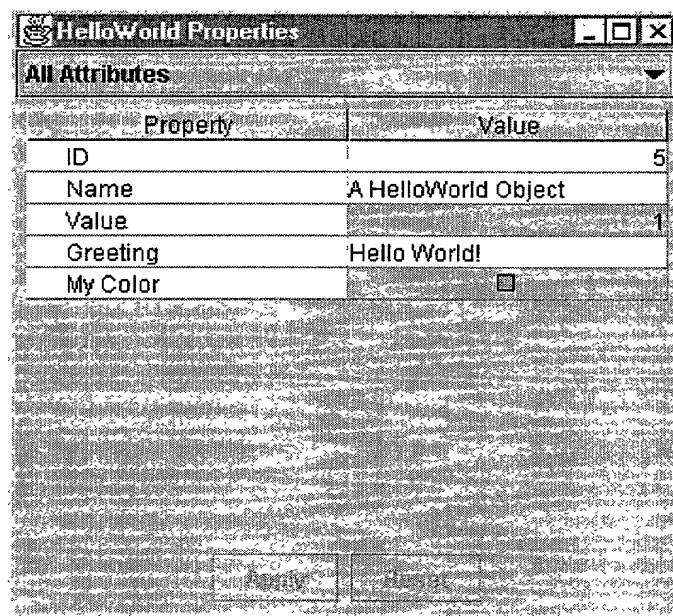
}

}
```


10039306 1906

1904

Figure 20



The screenshot shows a window titled 'HelloWorld Properties' with a standard Windows-style title bar (minimize, maximize, close buttons). Below the title bar is a tab labeled 'All Attributes'. The main content area contains a table with two columns: 'Property' and 'Value'. The table lists five attributes: ID (5), Name (A HelloWorld Object), Value (1), Greeting (Hello World!), and My Color (represented by a small square color swatch). The window has a dark, textured background.

Property	Value
ID	5
Name	A HelloWorld Object
Value	1
Greeting	Hello World!
My Color	

# Figure 21

## HelloWorld.java

```
/* XIS Tutorial standalone sequence step 3 data class. */
import java.awt.Color;
// (property change support moved to HelloWorldTranslator)

public class HelloWorld {
    private int value = 1;
    private Color myColor = Color.green;

    public String toString() {
        return "A HelloWorld Object";
    }

    public String getGreeting() {
        return "Hello World!";
    }

    public int getID() {
        return 5;
    }

    public int getValue() {
        return value;
    }

    /*{*/
    public void setValue(int value) {
        // all the worrying about change events is moved to the translator,
        // so we just need to do the bare change operation (unless nonXIS
        // components need to listen to PropertyChanges)
        this.value = value;
    }
    /*}*/

    public Color getMyColor() {
        return myColor;
    }

    /*{*/
    public void setMyColor(Color myColor) {
        // just need to set the value (see setValue())
        this.myColor = myColor;
    }
    /*}*/
}
```

## Figure 22A

### HelloWorldTranslator.java

```
/* XIS Tutorial standalone sequence step 3 data translator class. */
```

```
/*{*/
```

```
import com.xis.leif.im.AttributeGetRequest;  
import com.xis.leif.im.AttributeSetRequest;  
import com.xis.leif.im.Domain;  
import com.xis.leif.im.Translator;  
import com.xis.leif.im.FieldMetaData;  
import com.xis.domains.display.DisplayDomain;  
import com.xis.domains.movement.MovementDomain;  
import java.awt.Color;
```

```
public class HelloWorldTranslator extends Translator {
```

```
    // the domains from which canned attribute metadata will be taken  
    // NOTE, if an attribute appears in the methods below but its domain  
    // is NOT listed here, THE ATTRIBUTE WILL BE IGNORED BY XIS  
    private static final Domain[] baseDomains = new Domain[] {  
        DisplayDomain.getDomain(), MovementDomain.getDomain()  
    };
```

```
    // store info about the fields, such as whether they are preferred or not  
    private FieldMetaData[] fieldMetaDataArray;
```

```
    // Return the Domains that describe the Attributes.  
    public Domain[] getBaseDomains() {  
        return baseDomains;  
    }
```

```
    // this method returns info on each field defined in the methods below  
    public FieldMetaData[] getFieldMetaDataArray() {
```

```
        if (fieldMetaDataArray == null) {  
            // initialize default metadata  
            FieldMetaData dispname = new  
                FieldMetaData(DisplayDomain.displayName);  
            FieldMetaData pencolor = new  
                FieldMetaData(DisplayDomain.penColor);  
            FieldMetaData speed = new  
                FieldMetaData(MovementDomain.speed);  
            FieldMetaData course = new  
                FieldMetaData(MovementDomain.course);
```

## Figure 22B

### Continuation of HelloWorldTranslator.java

```
// attributes are visible ('preferred') by default; this
// turns this off for the course attribute
course.setVisibility(false);
// the order we put the attributes in here determines the order
// they appear in tables or property sheets
fieldMetaDataArray = new FieldMetaData[] {
    dispname, speed, course, pencolor
};
}

return fieldMetaDataArray;
}

////////////////////////////////////
// the following methods expose attributes of the HelloWorld class;
// instead of calling the class methods directly, XIS will access
// everything through this translator class
////////////////////////////////////
public String getDisplayName(AttributeGetRequest attributeGetRequest) {
    return ((HelloWorld)
        attributeGetRequest.getRawDataItem()).toString();
}

public Color getPenColor(AttributeGetRequest attributeGetRequest) {
    return ((HelloWorld)
        attributeGetRequest.getRawDataItem()).getMyColor();
}

public void setPenColor(AttributeSetRequest attributeSetRequest,
    Color penColor) {
    HelloWorld helloWorld = (HelloWorld)
        attributeSetRequest.getRawDataItem();
    Color oldPenColor = helloWorld.getMyColor();
    if (!penColor.equals(oldPenColor)) {
        helloWorld.setMyColor(penColor);
        // fire property change event to notify other XIS objects
        attributeSetRequest.getBaseDataItem().fireAttributeChanged(
            DisplayDomain.pencolor, oldPenColor, penColor, true);
    }
}

public double getSpeed(AttributeGetRequest attributeGetRequest) {
    return (double) ((HelloWorld)
        attributeGetRequest.getRawDataItem()).getValue();
}
```

2206 B

2210

10039306 102201



## Figure 22C

### Continuation of HelloWorldTranslator.java

```
public void setSpeed(AttributeSetRequest attributeSetRequest,
    double speed) {
    HelloWorld helloWorld = (HelloWorld)
        attributeSetRequest.getRawDataItem();
    Double oldSpeed = new Double(((double)helloWorld.getValue()));
    if (oldSpeed.doubleValue() != speed) {
        helloWorld.setValue((int)speed);
        // fire property change event to notify other XIS objects
        attributeSetRequest.getBaseDataItem().fireAttributeChanged(
            MovementDomain.speed, oldSpeed, new Double(speed), true);
    }
}
```

```
// this is a dummy attribute to demonstrate field metadata
public double getCourse(AttributeGetRequest attributeGetRequest) {
    return (double) 0;
}
```

```
/**
 * // uncomment this to allow reflection to expose additional attributes
 * // (see documentation under "Fooling Around")
 * // public HelloWorldTranslator() {
 * //     introspectExcept(new String[] {"value", "myColor"});
 * // }
 */
}
```

## Figure 23A

### TestHarness.java

/\* XIS Tutorial standalone sequence step 3 XIS interfacing. \*/

```
import javax.swing.JFrame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import com.xis.propertysheet.PropertySheetInfoBean;
import com.xis.ui.UIBeanEvent;
import com.xis.ui.UIBeanAdapter;
import com.xis.leif.im.BaseInfoModel;
import jclass.chart.JCChart;
import com.xis.plot.PlotInfoBean;
import com.xis.plot.chartviews.LeifChartView;
/*{*/
import com.xis.leif.im.TranslatorRegistry;
import com.xis.leif.im.LeifDataItem;
import com.xis.leif.im.InfoModel;
import com.xis.leif.im.BaseInfoModel;
import com.xis.domains.movement.MovementDomain;
import com.xis.domains.movement.MovementDomainWrapper;
import com.xis.leif.im.LeifDataItemDelegator;
import java.lang.reflect.InvocationTargetException;
import com.xis.leif.im.UndefinedLeifAttributeException;
/*}*/

public class TestHarness {

/*{*/
    protected static LeifDataItem leifHello;

    static {
        // Register the translator for HelloWorld. In fact this is really
        // only necessary when we have not followed the standard naming
        // convention (see docs), but it can't hurt.
        TranslatorRegistry.getTranslatorRegistry().registerObjectSchema(
            HelloWorld.class, HelloWorldTranslator.class);
    }
/*}*/

    public static void main(String[] args) {

        // the plugin manager is only required for more complex applications
        // involving multiple components integrated at runtime
        BaseInfoModel.setStartingPluginManager(false);

        HelloWorld hello = new HelloWorld();
```

} 2302

10039300 100201

## Figure 23B

### Continuation of TestHarness.java

```
// a property sheet infobean to display HelloWorld's attributes
PropertySheetInfoBean properties = new PropertySheetInfoBean();
properties.addRawDataItem(hello);

// add a listener for 'OK' or 'cancel', which generate 'close' events
properties.addUIBeanListener(
    new UIBeanAdapter() {
        public void closed(UIBeanEvent event) {
            System.exit(0);
        }
    }
);

// a top-level frame to hold our property sheet infobean
JFrame propertySheetFrame = new JFrame("HelloWorld Properties");
// add a listener for window closing
propertySheetFrame.addWindowListener(
    new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    }
);

// stick the property Sheet bean in the frame and display it
propertySheetFrame.getContentPane().add(properties);
propertySheetFrame.pack();
propertySheetFrame.setVisible(true);

// now we create a plot infobean to plot HelloWorld's numeric attribute
PlotInfoBean plot = new PlotInfoBean();
plot.addRawDataItems(new Object[] { hello });
plot.setChartType(JCChart.BAR);
// the alternatives are SCATTER_PLOT, PLOT, AREA, PIE, CANDLE,
// and STACKING_BAR, though not all will make sense in this example

// We can set the attribute for initial display on the plot;
// if we do, this must consist of the attribute name preceded
// by the fully-qualified classname which ORIGINALLY DEFINES
// the attributeDescriptor -- i.e., using "HelloWorld.speed"
// here will NOT work! If the descriptor is not defined in a
// domain or translator class, then it will have been defined
// dynamically through introspection when the first instance
// of the data item is dropped into an XIS InfoBean.
plot.setYAxisAttribute(
    "com.xis.domains.movement.MovementDomain.speed");
```

## Figure 23C

### Continuation of TestHarness.java

```
plot.setDynamicAdjustment(true); // so axes track value magnitude
plot.setBarChartAdjusting(true); // needed in some cases for bar chart
```

```
// a top-level frame as before to hold our plot infobean
JFrame plotFrame = new JFrame("HelloWorld Plot");
```

```
// stick the plot bean in and put it up
plotFrame.getContentPane().add(plot);
plotFrame.pack();
plotFrame.setVisible(true);
```

```
/*  
// create a leifDataItem version of hello and start a thread that  
// will increase it  
leifHello =  
    BaseInfoModel.getBaseInfoModel().getLeifDataItem(hello);  
    new Accelerate();  
*/
```

```
    } // main  
  
}
```

```
/*  
// thread to update the speed attribute on the leifHello instance we created  
class Accelerate extends Thread {
```

```
    public Accelerate() {  
        super("Accelerator Thread");  
        start();  
    }
```

```
    public void run() {
```

```
        // wrap the LeifDataItem leifHello in a convenience wrapper that  
        // gives access to attributes within that domain, if they exist  
        MovementDomainWrapper helloMovementWrapper =  
            MovementDomain.takeWrapper(TestHarness.leifHello);
```

## Figure 23D

### Continuation of TestHarness.java

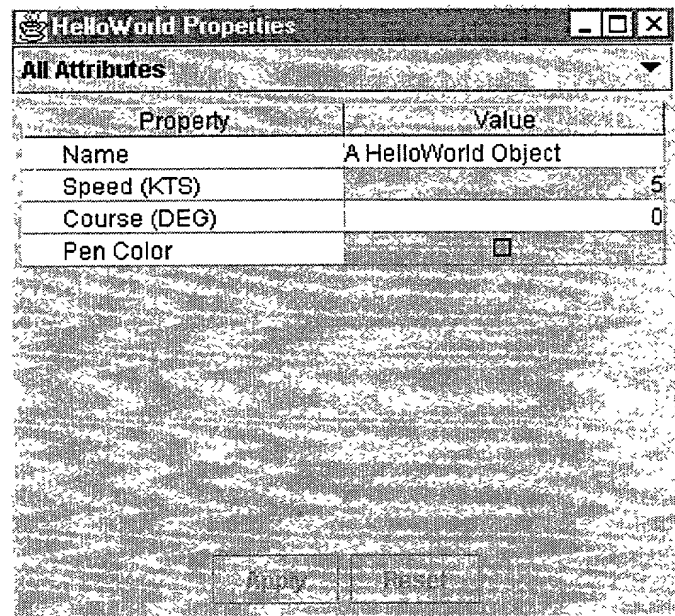
```
while (true) {

    // sleep for 0.5 seconds, then...
    try {
        sleep(500);
    } catch (InterruptedException e) {
        System.exit(1);
    }

    // ..update the speed attribute
    try {
        helloMovementWrapper.setSpeed(
            helloMovementWrapper.getSpeed()+1);
    } catch (UndefinedLeifAttributeException ulae) {
        // exception if this data item doesn't have this attribute
        System.exit(1); // usually we would do something better
    } catch (InvocationTargetException ite) {
        // sweep up any exception tossed by the underlying raw item
        System.exit(1); // usually we would do something better
    }
} // while
} // run()
};
/* */
```

10039306 10000

Figure 24A




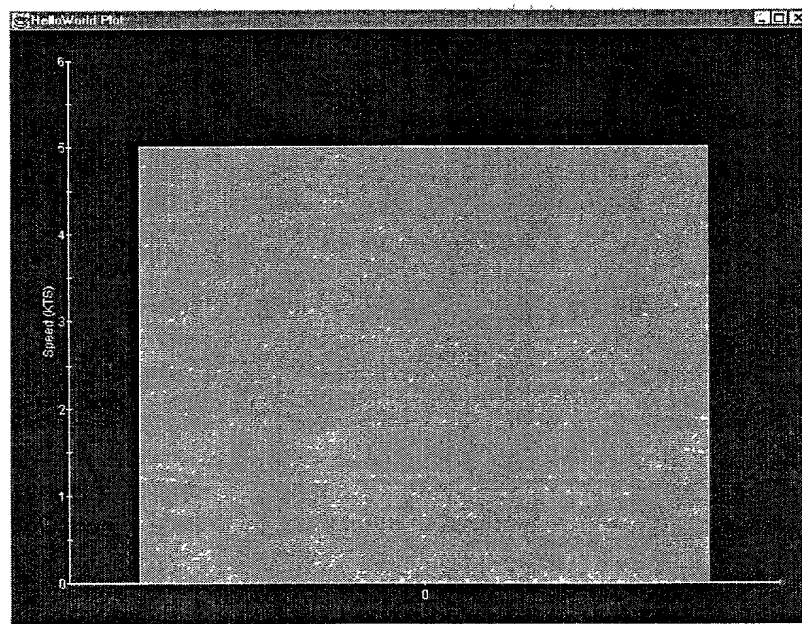
Property	Value
Name	A HelloWorld Object
Speed (KTS)	5
Course (DEG)	0
Pen Color	

Figure 24B



## Figure 25A

### HelloWorld.java

/\* XIS Tutorial standalone sequence step 4 data class. \*/

import java.awt.Color;

/\*{\*/

import com.xis.leif.im.FieldMetaData;

import com.xis.domains.display.DisplayDomain;

import com.xis.domains.movement.MovementDomain;

import com.xis.leif.im.Domain;

import com.xis.leif.im.AttributeGetRequest;

import com.xis.leif.im.AttributeSetRequest;

import com.xis.leif.im.AttributeDescriptor;

import java.beans.PropertyChangeSupport;

import java.beans.PropertyChangeListener;

/\*}\*/

public class HelloWorld {

private int value = 1;

private Color myColor = Color.green;

/\*{\*/

public static AttributeDescriptor getDisplayNameDescriptor() {  
return DisplayDomain.displayName;

}  
public static AttributeDescriptor getSpeedDescriptor() {  
return MovementDomain.speed;

}  
public static AttributeDescriptor getPenColorDescriptor() {  
return DisplayDomain.penColor;

}

/\*}\*/

/\*{\*/ // this property change support code as in step 2 /\*}\*/

// this member class helps distribute property change events within XIS  
private PropertyChangeSupport propertyChangeSupport =  
new PropertyChangeSupport(this);

// two aux methods to let other XIS objects pay attention to this one

public void addPropertyChangeListener(PropertyChangeListener l) {  
propertyChangeSupport.addPropertyChangeListener(l);

}  
public void removePropertyChangeListener(PropertyChangeListener l) {  
propertyChangeSupport.removePropertyChangeListener(l);

}

2502

2504

## Figure 25B

### Continuation of HelloWorld.java

```
public String toString() {
    return "A HelloWorld Object";
}

public String getGreeting() {
    return "Hello World!";
}

public int getID() {
    return 5;
}

public int getValue() {
    return value;
}

/*{*/ // this function as in step 2 /*}*/
public void setValue(int value) {
    // we only want to update and fire property change if really changes
    if (this.value != value) {
        int oldValue = this.value;
        this.value = value;
        // fire property change event to notify other XIS objects
        propertyChangeSupport.firePropertyChange("value", oldValue, value);
    }
}

/*{*/
// "myColor"-related methods changed to expose "penColor" instead

public Color getPenColor() {
    return myColor;
}

public void setPenColor(Color penColor) {
    // we only want to update and fire property change if really changes
    if (penColor != this.myColor) {
        Color oldPenColor = this.myColor;
        this.myColor = penColor;
        // fire property change event to notify other XIS objects
        propertyChangeSupport.firePropertyChange("penColor",
            oldPenColor, penColor);
    }
}
```



## Figure 25C

### Continuation of HelloWorld.java

```
// expose toString() return under a new name
public String getDisplayName() {
    return toString();
}

// expose "value" under a new name
public double getSpeed() {
    return (double) getValue();
}

public void setSpeed(double speed) {
    // we only want to update and fire property change if really changes
    Double oldSpeed = new Double((double)this.getValue());
    if (speed != oldSpeed.doubleValue()) {
        setValue((int)speed);
        // fire property change event to notify other XIS objects
        propertyChangeSupport.firePropertyChange("speed", oldSpeed,
            new Double(speed));
    }
}

/* */
/* */
// store info about the fields, such as whether they are preferred or not
private static FieldMetaData[] fieldMetaDataArray;

// this method returns info on each field defined in the methods below
public static FieldMetaData[] getFieldMetaDataArray() {
    if (fieldMetaDataArray == null) {
        // initialize default metadata
        FieldMetaData dispname = new
            FieldMetaData(DisplayDomain.displayName);
        FieldMetaData pencolor = new
            FieldMetaData(DisplayDomain.penColor);
        FieldMetaData speed = new
            FieldMetaData(MovementDomain.speed);
        // could customize the field metadata here
        fieldMetaDataArray = new FieldMetaData[] {
            dispname, speed, pencolor
        };
    }
    return fieldMetaDataArray;
}

/* */
}
```

2506

Figure 26

HelloWorld Properties

All Attributes

Property	Value
Name	A HelloWorld Object
Speed (KTS)	20
Pen Color	<input type="checkbox"/>
Value	20
Greeting	Hello World!
ID	5

Apply Reset

FOUO 9066001

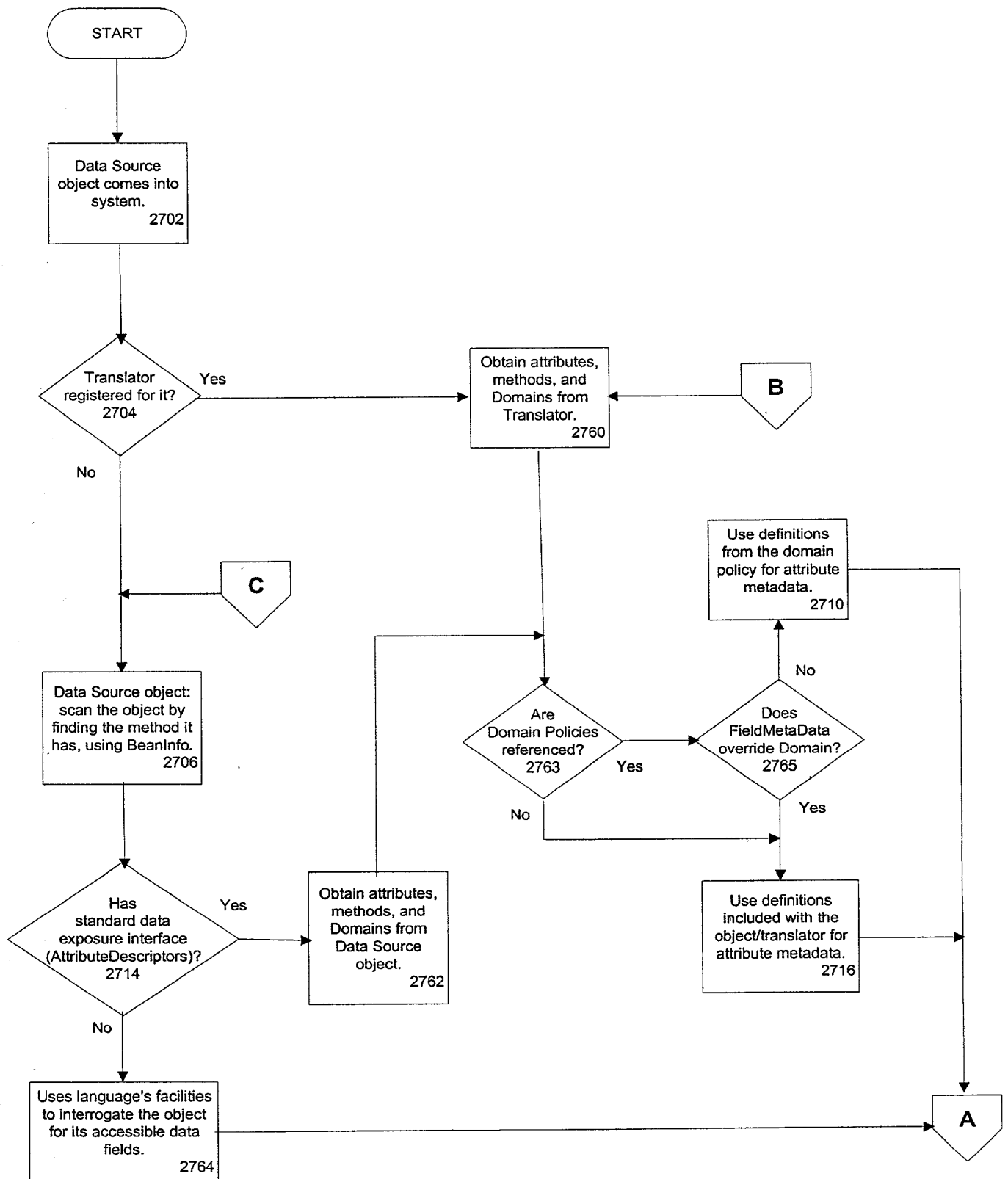


Figure 27A

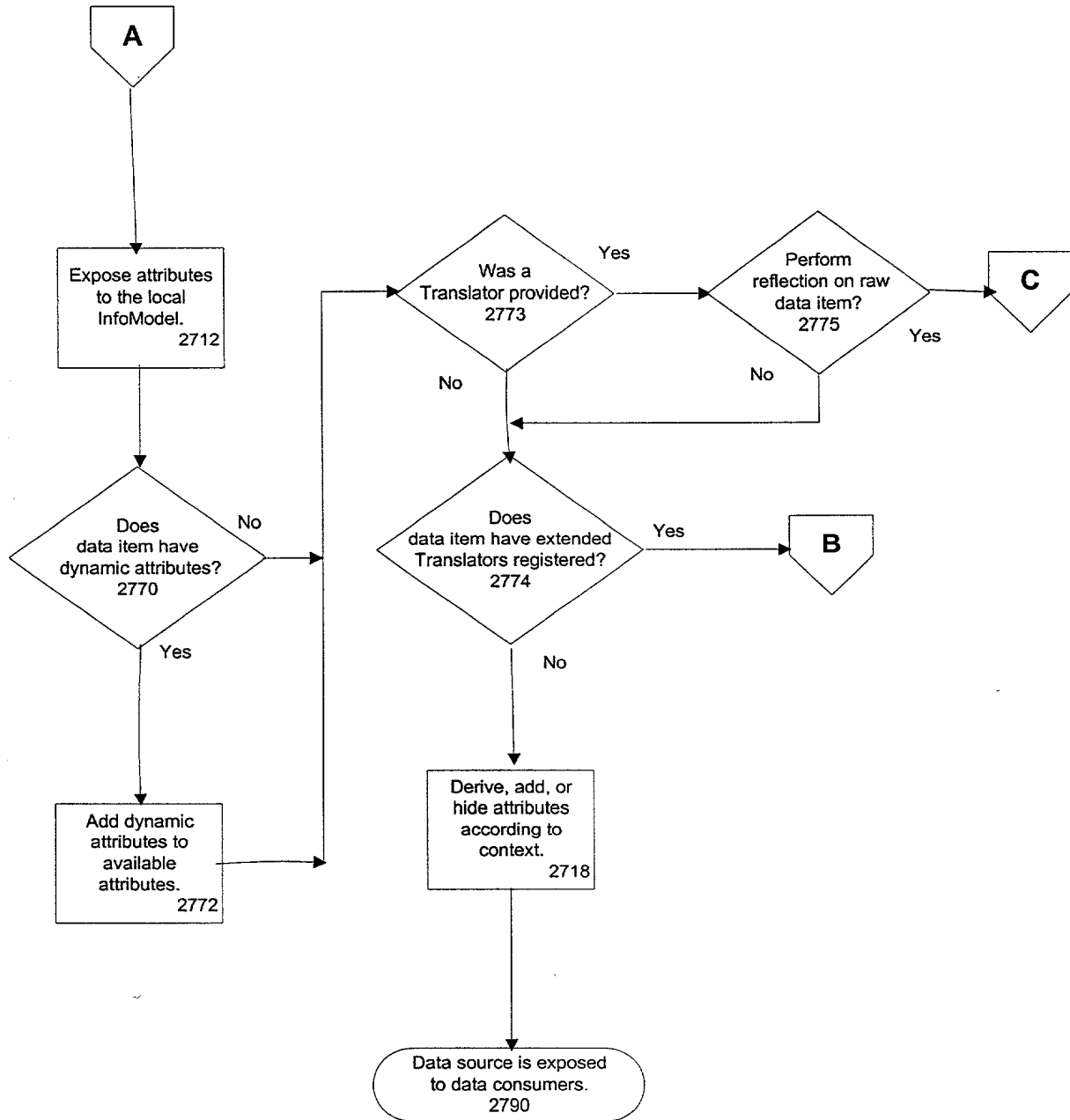


Figure 27B

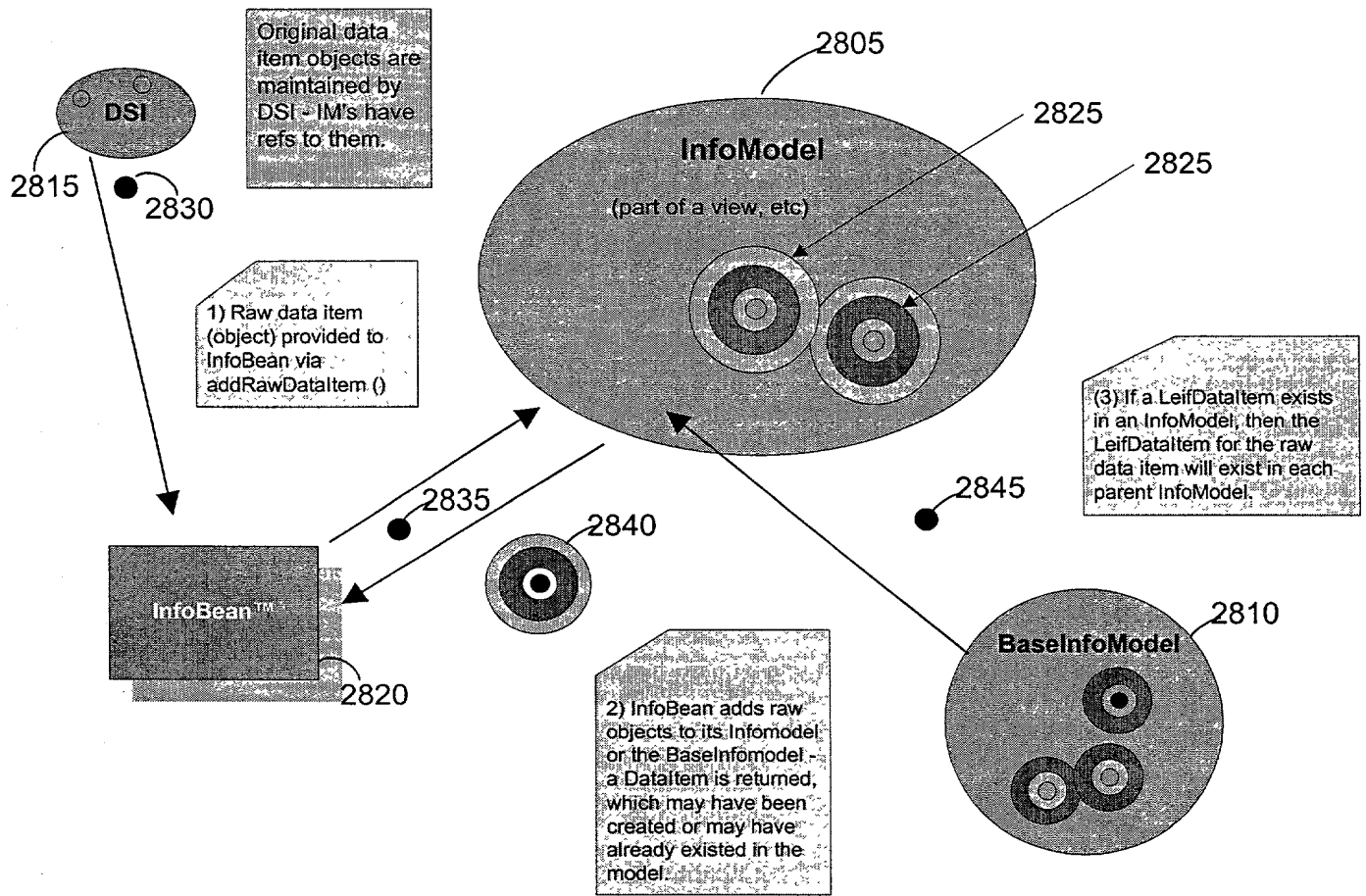


Figure 28

1009306 10001

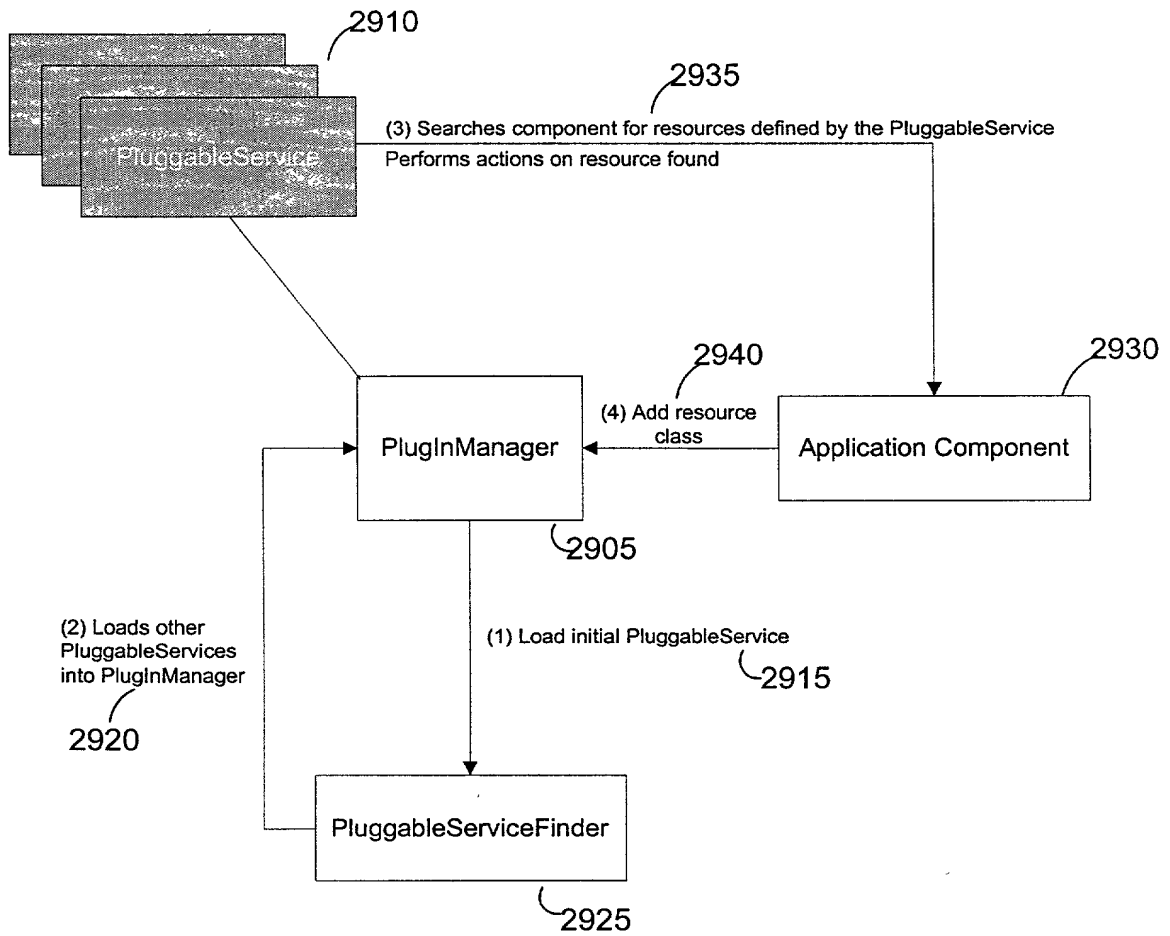


Figure 29

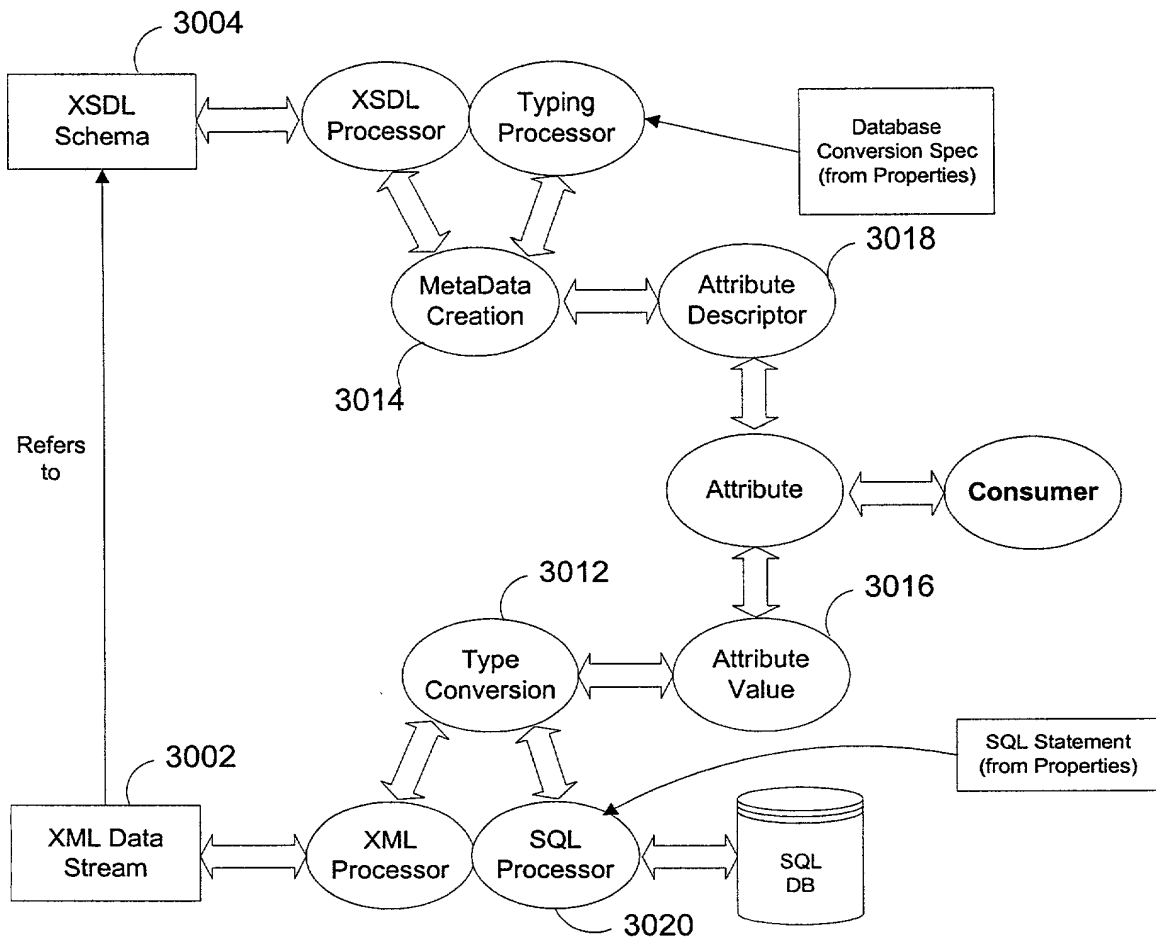


Figure 30

100300 100300

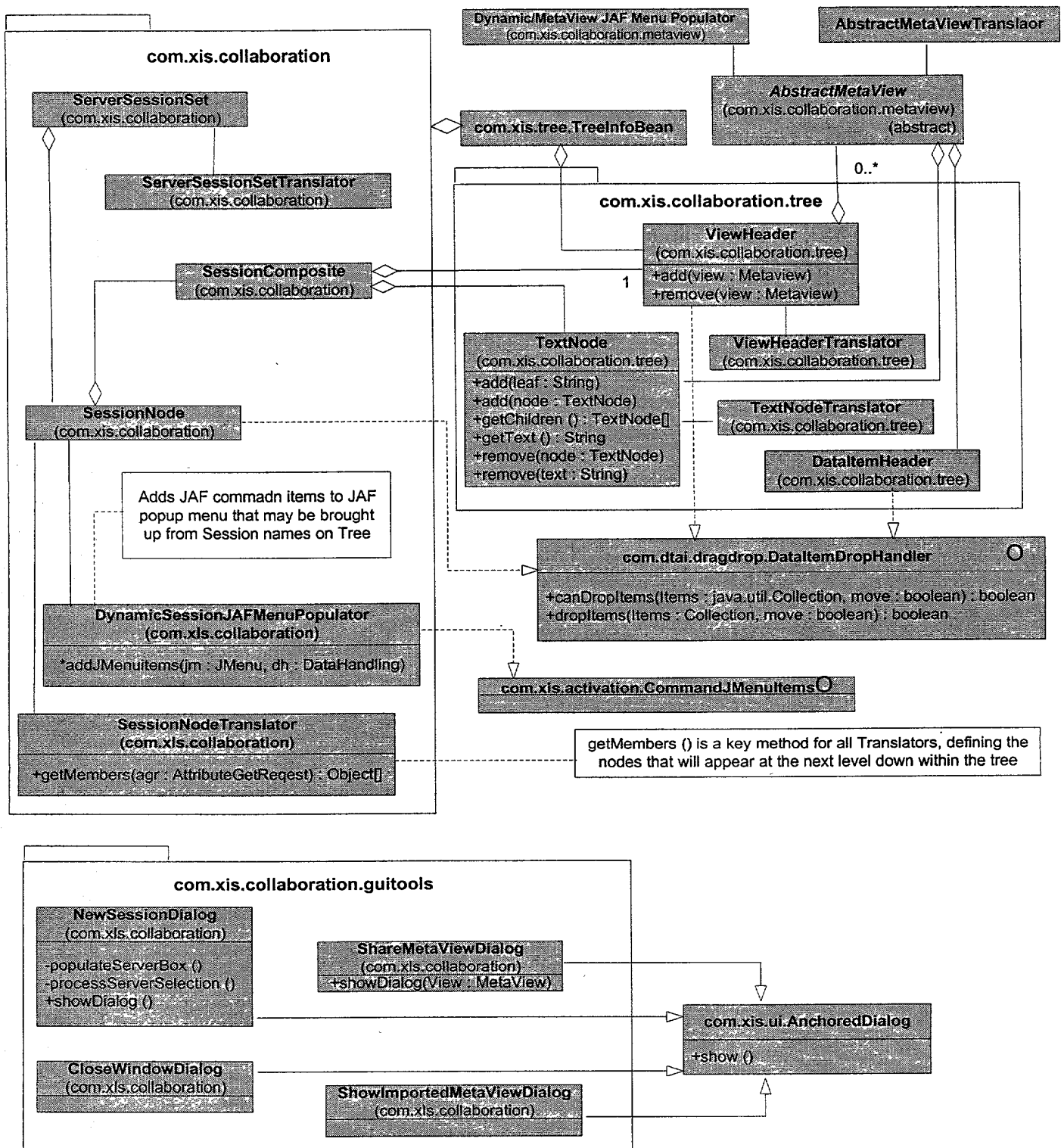


Figure 31



10039306-10201

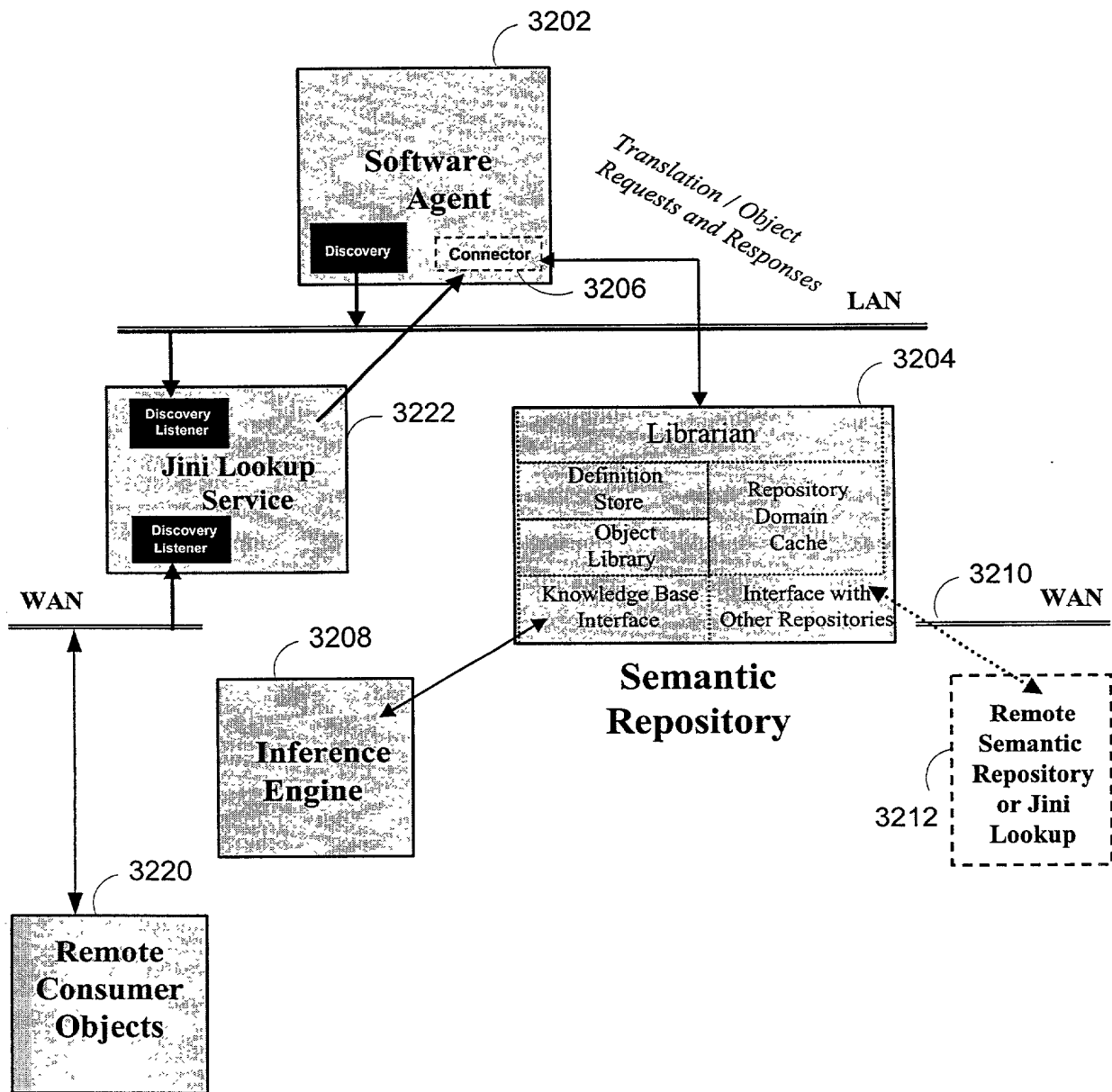


Figure 32

## Class ContentInfoBean

```

java.lang.Object
|
+--java.awt.Component
|   |
|   +--java.awt.Container
|       |
|       +--javax.swing.JComponent
|           |
|           +--javax.swing.JPanel
|               |
|               +--com.xis.ui.AbstractUIBean
|                   |
|                   +--com.xis.leif.infobeans.DataItemSinkUIBean
|                       |
|                       +--com.xis.infobeans.content.ContentInfoBean

```

### All Implemented Interfaces:

Accessible, BeanContextChildOwner, BeanContextChildOwnerDelegator, BeanContextProxy,  
BeanContextServicesOwnerDelegator, ClipboardUser, DataItemSink, ImageObserver, MenuContainer,  
Serializable, StateSavable, UIBean

---

```

public class ContentInfoBean
extends DataItemSinkUIBean
implements ClipboardUser

```

The ContentInfoBean class is a visual component that displays the contents of a raw data item. If no contents are available, it defaults to a split pane containing the JAF menu and the PropertySheet of the raw data item. The contents may have be multipart, and may be text, html, rich text, or an image. Multimedia support will soon be added.

### Author:

Jaime Garcia, Polaxis, Inc.

### See Also:

Serialized Form

---

Inner classes inherited from class javax.swing.JPanel
---

<u>JPanel.AccessibleJPanel</u>
--------------------------------

Inner classes inherited from class javax.swing.JComponent
---

<u>JComponent.AccessibleJComponent</u>
--

Inner classes inherited from class java.awt.Container
---

<u>Container.AccessibleAWTContainer</u>
---

Inner classes inherited from class `java.awt.Component`

`Component.AccessibleAWTComponent`

## Field Summary

<code>static ResourceBundle</code>	<code>RESOURCES</code> localized resources for this view object.
------------------------------------	---

Fields inherited from class `com.xis.ui.AbstractUIBean`

`uibeanListener`

Fields inherited from class `javax.swing.JComponent`

`accessibleContext`, `listenerList`, `TOOL_TIP_TEXT_KEY`, `ui`, `UNDEFINED_CONDITION`,  
`WHEN_ANCESTOR_OF_FOCUSED_COMPONENT`, `WHEN_FOCUSED`, `WHEN_IN_FOCUSED_WINDOW`

Fields inherited from class `java.awt.Component`

`BOTTOM_ALIGNMENT`, `CENTER_ALIGNMENT`, `LEFT_ALIGNMENT`, `RIGHT_ALIGNMENT`, `TOP_ALIGNMENT`

Fields inherited from interface `java.awt.image.ImageObserver`

`ABORT`, `ALLBITS`, `ERROR`, `FRAMEBITS`, `HEIGHT`, `PROPERTIES`, `SOMEBITS`, `WIDTH`

## Constructor Summary

<code>ContentInfoBean()</code> Default constructor that creates an empty <code>ContentInfoBean</code> .
--

## Method Summary

<code>void</code>	<code>addRawDataItem(Object rawDataItem)</code> Load the raw data item into the <code>ContentInfoBean</code> .
<code>void</code>	<code>addRawDataItems(Object[] rawDataItems)</code> Add the raw data items in the array.
<code>boolean</code>	<code>canClear()</code> Return true if the <code>ContentInfoBean</code> has an object and it is selected
<code>boolean</code>	<code>canClear(Object[] items)</code> Return true if the specified items can be cleared.
<code>boolean</code>	<code>canCopy()</code> Return true if the <code>ContentInfoBean</code> has an object and it is selected
<code>boolean</code>	<code>canCopy(Object[] items)</code> Return true if the specified items can be copied.
<code>boolean</code>	<code>canCut()</code> Return true if the <code>ContentInfoBean</code> has an object and it is selected
<code>boolean</code>	<code>canCut(Object[] items)</code> Return true if the specified items can be cut.
<code>boolean</code>	<code>canPaste()</code> Return true if the <code>ContentInfoBean</code> can paste new objects, false if not.
<code>boolean</code>	<code>canSelectAll()</code> Return true if the <code>ContentInfoBean</code> can select all objects.

boolean	<u>canSelectNone()</u> Return true if the ContentInfoBean can un-select all objects.
void	<u>clear()</u> Notify the ContentInfoBean to remove the current raw data item only if it is selected.
void	<u>clear(Object[] items)</u> Clears the given items.
void	<u>clearAll()</u> Removes the currently loaded object.
boolean	<u>contains(Object[] items)</u> Return true if this ContentInfoBean contains <i>all</i> the objects of the given array.
boolean	<u>containsComponent(Component component)</u> Check if the given component is contained by this InfoBean
void	<u>copy(Clipboard clipboard)</u> Called to invoke this ContentInfoBean's copy action, which is to copy all selected data to the Clipboard.
void	<u>copy(Clipboard clipboard, Object[] items)</u> Copies the given items into the Clipboard.
protected void	<u>createContent()</u> Method called when there is no content to display for a raw data item.
void	<u>cut(Clipboard clipboard)</u> Cut selected items from the ContentInfoBean and post them into Clipboard.
void	<u>cut(Clipboard clipboard, Object[] items)</u> Cut the given items from the ContentInfoBean and post them into the given Clipboard only if they occur in the ContentInfoBean.
protected Container	<u>getContainerForContent(int index)</u> Get a container with the contents of the content object at the given index, or null if the content type is not supported.
Object	<u>getContents()</u> Fetch the currently loaded raw data item
JAFAndPropertyComponent	<u>getJAFAndPropertyComponent()</u> Get the JAFAndProperty component used by the ContentInfoBean to display the contents for raw data items that have nothing else to display.
JMenu	<u>getLeifDataItemMenu(LeifDataItem dataItem, boolean showCutPasteItems)</u> Return the data item menu for a LeifDataItem (usually the selected LeifDataItem).
TypedResourceBundle	<u>getResources()</u> Return the ResourceBundle for this ContentInfoBean.
Object[]	<u>getSelectedObjects()</u> Get an array of selected objects.
void	<u>infoModelChanged()</u> Messaged to indicate an InfoModel change for this InfoBean or one or more of its LeifDataItems.
boolean	<u>isCreatingContent()</u> Check whether default content creation is set.
boolean	<u>isDragEnabled()</u> Return true if the default Drag support is enabled.
boolean	<u>isDropEnabled()</u> Return true if the default Drop support is enabled.

boolean	<code>isSelected()</code> Check the selected state of the content object, if there is currently one loaded.
boolean	<code>isXISNotifying()</code> Check whether the ContentInfoBean is updating based on XIS events and is notifying XIS of raw data item attribute changes
void	<code>paste(Clipboard clipboard)</code> Paste the data Objects from the given clipboard.
void	<code>removeAllRawDataItems()</code> Remove all of the raw data items that are currently loaded
void	<code>removeRawDataItem(Object rawDataItem)</code> Remove the given raw data item if it is the currently loaded raw data item
void	<code>removeRawDataItems(Object[] rawDataItems)</code> Remove the raw data items in the array.
void	<code>selectAll()</code> Set the selection state of the object to true.
void	<code>selectNone()</code> Set the selection state of the object to false.
void	<code>setCreateContent(boolean create)</code> Set whether content should be created for objects that do not have any displayable content, via a JAFAndPropertyComponent.
void	<code>setDragEnabled(boolean enabledrag)</code> Set the status of the default Drag support.
void	<code>setDragOwnerProxy(DragOwner dragProxy)</code> Set a DragOwner "proxy" for this InfoBean.
void	<code>setDropEnabled(boolean enabledrop)</code> Set the status of the default Drop support.
void	<code>setDropOwnerProxy(DropOwner dropProxy)</code> Set a DropOwner "proxy" for this InfoBean.
void	<code>setSelection(boolean selected)</code> Set the selection state of the content object
void	<code>setXISNotifying(boolean notify)</code> Set whether the ContentInfoBean should update based on XIS events and should notify XIS of raw data item attribute changes

#### Methods inherited from class com.xis.leif.infobeans.DataItemSinkUIBean

`addJAFPopulator, addRawDataItemsAsGroup, addService, close, createBeanContextServicesOwnerDelegator, dispose, getBeanContextProxy, getBeanContextServices, getEzContext, getInfoModel, getJAFPopulators, getLeifDataItemMenu, getLeifDataItemMenu, getLeifDataItemMenu, getMenuBar, getOwnedBeanContextChild, getService, getService, getService, getStatusBars, getToolBars, getUIs, initializeBeanContextResources, initializeBeanContextServices, invalidateInfoModel, isDropPasteEnabled, isHandlingClipboardOperations, releaseBeanContextResources, releaseBeanContextServices, revokeAnchoredDialogProvider, revokeFrameProvider, revokeService, setDropPasteEnabled, setHandlingClipboardOperations, validatePendingSetBeanContext`

#### Methods inherited from class com.xis.ui.AbstractUIBean

`addUIBeanListener, finalize, getShortTitle, getTitle, getUIComponents, isActive, isClosed, isCloseOK, processUIBeanEvent, removeUIBeanListener, restoreState, saveState, setActive, setShortTitle, setTitle`

getAccessibleContext, getUIClassID, paramString, updateUI

[addAncestorListener](#), [addNotify](#), [addPropertyChangeListener](#), [addPropertyChangeListener](#), [addVetoableChangeListener](#), [computeVisibleRect](#), [contains](#), [createToolTip](#), [disable](#), [enable](#), [firePropertyChange](#), [firePropertyChange](#), [firePropertyChange](#), [firePropertyChange](#), [firePropertyChange](#), [firePropertyChange](#), [firePropertyChange](#), [firePropertyChange](#), [firePropertyChange](#), [firePropertyChange](#), [fireVetoableChange](#), [getActionForKeyStroke](#), [getActionMap](#), [getAlignmentX](#), [getAlignmentY](#), [getAutoscrolls](#), [getBorder](#), [getBounds](#), [getClientProperty](#), [GetComponentGraphics](#), [getConditionForKeyStroke](#), [getDebugGraphicsOptions](#), [getGraphics](#), [getHeight](#), [getInputMap](#), [getInputMap](#), [getInputVerifier](#), [getInsets](#), [getInsets](#), [getListeners](#), [getLocation](#), [getMaximumSize](#), [getMinimumSize](#), [getNextFocusableComponent](#), [getPreferredSize](#), [getRegisteredKeyStrokes](#), [getRootPane](#), [getSize](#), [getToolTipLocation](#), [getToolTipText](#), [getToolTipText](#), [getTopLevelAncestor](#), [getVerifyInputWhenFocusTarget](#), [setVisibleRect](#), [getWidth](#), [getX](#), [getY](#), [grabFocus](#), [hasFocus](#), [hide](#), [isDoubleBuffered](#), [isFocusCycleRoot](#), [isFocusTraversable](#), [isLightweightComponent](#), [isManagingFocus](#), [isMaximumSizeSet](#), [isMinimumSizeSet](#), [isOpaque](#), [isOptimizedDrawingEnabled](#), [isPaintingTile](#), [isPreferredSizeSet](#), [isRequestFocusEnabled](#), [isValidatedRoot](#), [paint](#), [paintBorder](#), [paintChildren](#), [paintComponent](#), [paintImmediately](#), [paintImmediately](#), [print](#), [printAll](#), [printBorder](#), [printChildren](#), [printComponent](#), [processComponentKeyEvent](#), [processFocusEvent](#), [processKeyBinding](#), [processKeyEvent](#), [processMouseEvent](#), [putClientProperty](#), [registerKeyboardAction](#), [registerKeyboardAction](#), [removeAncestorListener](#), [removeNotify](#), [removePropertyChangeListener](#), [removePropertyChangeListener](#), [removeVetoableChangeListener](#), [repaint](#), [repaint](#), [requestDefaultFocus](#), [requestFocus](#), [resetKeyboardActions](#), [reshape](#), [revalidate](#), [scrollRectToVisible](#), [setActionMap](#), [setAlignmentX](#), [setAlignmentY](#), [setAutoscrolls](#), [setBackground](#), [setBorder](#), [setDebugGraphicsOptions](#), [setDoubleBuffered](#), [setEnabled](#), [setFont](#), [setForeground](#), [setInputMap](#), [setInputVerifier](#), [setMaximumSize](#), [setMinimumSize](#), [setNextFocusableComponent](#), [setOpaque](#), [setPreferredSize](#), [setRequestFocusEnabled](#), [setToolTipText](#), [setUI](#), [setVerifyInputWhenFocusTarget](#), [setVisible](#), [unregisterKeyboardAction](#), [update](#)

add, add, add, add, add, addContainerListener, addImpl, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getLayout, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setLayout, validate, validateTree

```

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener,
addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener,
addMouseMotionListener, bounds, checkImage, checkImage, coalesceEvents, contains,
createImage, createImage, disableEvents, dispatchEvent, enable, enableEvents,
enableInputMethods, getBackground, getBounds, getColorModel, getComponentOrientation,
getCursor, getDropTarget, getFont, getFontMetrics, getForeground,
getGraphicsConfiguration, getInputContext, getInputMethodRequests, getLocale,
getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getToolkit,
getTreeLock, gotFocus, handleEvent, imageUpdate, inside, isDisplayable, isEnabled,
isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location,
lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move,
nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent,
processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent,
processMouseEvent, remove, removeComponentListener, removeFocusListener,
removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener,
removeKeyListener, removeMouseListener, removeMouseMotionListener, repaint, repaint,
repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor,
setDropTarget, setLocale, setLocation, setLocation, setName, setSize, setSize, show,
show, size, toString, transferFocus

```

```
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait
```

Methods inherited from interface `com.xis.ui.ClipboardUser`

isHandlingClipboardOperations, setHandlingClipboardOperations

### Methods inherited from interface com.xis.ui.UIBean

```
addChangeListener, removeChangeListener
```

### Field Detail

## RESOURCES

```
public static final TypedResourceBundle RESOURCES
```

localized resources for this view object.

## Constructor Detail

## ContentInfoBean

```
public ContentInfoBean()
```

Default constructor that creates an empty ContentInfoBean.

## Method Detail

## getContents

```
public Object getContents()
```

Fetch the currently loaded raw data item

**Returns:**

the currently loaded object, or null if no object is loaded

## setXISNotifying

```
public void setXISNotifying(boolean notify)
```

Set whether the ContentInfoBean should update based on XIS events and should notify XIS of raw data item attribute changes

### Parameters:

notify - if true then notify XIS, else do not

**isXISNotifying**

```
public boolean isXISNotifying()
```

Check whether the ContentInfoBean is updating based on XIS events and is notifying XIS of raw data item attribute changes

Returns:

true if it is notifying XIS, else false

---

**removeAllRawDataItems**

```
public void removeAllRawDataItems()
```

Remove all of the raw data items that are currently loaded

---

**getResources**

```
public TypedResourceBundle getResources()
```

Return the ResourceBundle for this ContentInfoBean..

Overrides:

getResources in class DataItemSinkUIBean

Returns:

the statically sourced ResourceBundle.

---

**infoModelChanged**

```
public void infoModelChanged()
```

Messaged to indicate an InfoModel change for this InfoBean or one or more of its LeafDataItems. This should reload all currently loaded data items to pick up InfoModel changes.

Overrides:

infoModelChanged in class DataItemSinkUIBean

---

**addRawDataItems**

```
public void addRawDataItems(Object[] rawDataItems)
```

Add the raw data items in the array. Will only add if the array has only one object and the object is not the currently loaded rawDataItem.

Parameters:

rawDataItems - the array of objects to be added

---

**addRawDataItem**

```
public void addRawDataItem(Object rawDataItem)
```



rawDataItem - the raw object to display

```
public void removeRawDataItem(Object rawDataItem)
```

Remove the given raw data item if it is the currently loaded raw data item

Overrides:

`removeRawDataItem` in class `DataItemSinkUIBean`

Parameters:

`rawDataItem` - the object to remove

### `getJAFAndPropertyComponent`

```
public JAFAndPropertyComponent getJAFAndPropertyComponent()
```

Get the JAFAndProperty component used by the ContentInfoBean to display the contents for raw data items that have nothing else to display.

Returns:

the current JAFAndPropertyComponent being used

### `getContainerForContent`

```
protected Container getContainerForContent(int index)
```

Get a container with the contents of the content object at the given index, or null if the content type is not supported. Subclass this if you need support for a type that is not already supported.

Parameters:

`index` - the index of in the object

Returns:

a Container with the contents at the given index

### `getSelectedObjects`

```
public Object[] getSelectedObjects()
```

Get an array of selected objects. This will return an empty array if there are no selected objects. If the raw data item is selected it will return an array of size 1 with the raw data item inside

Returns:

an array of selected objects

### `isSelected`

```
public boolean isSelected()
```

Check the selected state of the content object, if there is currently one loaded. If there is none, return false.

Returns:

true if there is an object and it is selected

### `selectAll`

```
public void selectAll()
```

Set the selection state of the object to true.

Specified by:

selectAll in interface ClipboardUser

Overrides:

selectAll in class DataItemSinkUIBean

## selectNone

```
public void selectNone()
```

Set the selection state of the object to false.

## setSelection

```
public void setSelection(boolean selected)
```

Set the selection state of the content object

Parameters:

the - new selection state

## canClear

```
public boolean canClear()
```

Return true if the ContentInfoBean has an object and it is selected

Specified by:

canClear in interface ClipboardUser

Overrides:

canClear in class DataItemSinkUIBean

Returns:

true if the raw data item is selected

See Also:

getSelectedObjects()

## canClear

```
public boolean canClear(Object[] items)
```

Return true if the specified items can be cleared. If the item is not in the ContentInfoBean return false. If there is more than one item return false.

Specified by:

canClear in interface ClipboardUser

Overrides:

canClear in class DataItemSinkUIBean

Parameters:

items - the Array of items to be cut.

Returns:

true if all of the items are present, otherwise false.

See Also:

contains(Object[])

---

### canCopy

public boolean canCopy()

Return true if the ContentInfoBean has an object and it is selected

Specified by:

canCopy in interface ClipboardUser

Overrides:

canCopy in class DataItemSinkUIBean

Returns:

true if the raw data item is selected

See Also:

canClear()

---

### canCopy

public boolean canCopy(Object[] items)

Return true if the specified items can be copied. If the item is not in the ContentInfoBean return false. If there is more than one item return false.

Specified by:

canCopy in interface ClipboardUser

Overrides:

canCopy in class DataItemSinkUIBean

Parameters:

items - the Array of items to be cut.

Returns:

true if all of the items are present, otherwise false.

See Also:

canClear(Object[])

---

### canCut

public boolean canCut()

Return true if the ContentInfoBean has an object and it is selected

Specified by:

canCut in interface ClipboardUser

Overrides:

canCut in class DataItemSinkUIBean

Returns:

true if the raw data item is selected

See Also:

canClear()

---

### canCut

public boolean canCut(Object[] items)

1005306 102031

Specified by:

**Overrides:**

### Parameters:

**Returns:**

**See Also:**

canPaste

Specified by:

**Overrides:**

**Returns:**

**canSelectAll**

Specified by:

**Overrides:**

**Returns:**

**canSelectNone**

**Returns:**

**clear**

```
public void clear()
```

Notify the ContentInfoBean to remove the current raw data item only if it is selected.

Specified by:

clear in interface ClipboardUser

Overrides:

clear in class DataItemSinkUIBean

---

## clear

```
public void clear(Object[] items)
```

Clears the given items. These items only get cleared if they actually occur in the ContentInfoBean.

Specified by:

clear in interface ClipboardUser

Overrides:

clear in class DataItemSinkUIBean

Parameters:

items - the items to cleared

---

## clearAll

```
public void clearAll()
```

Removes the currently loaded object.

---

## copy

```
public void copy(Clipboard clipboard)
```

Called to invoke this ContentInfoBean's copy action, which is to copy all selected data to the Clipboard.

Specified by:

copy in interface ClipboardUser

Overrides:

copy in class DataItemSinkUIBean

Parameters:

clipboard - the Clipboard object that gets posted to. The actual items posted are contained in a LeifTransferable.

---

## copy

```
public void copy(Clipboard clipboard,  
                Object[] items)
```

Copies the given items into the Clipboard..

Specified by:

copy in interface ClipboardUser

Overrides:

copy in class DataItemSinkUIBean

Parameters:

clipboard - the Clipboard object that gets posted to. The actual items posted are contained in a

LeifTransferable.  
 items - the array of Object items to copied.

---

### cut

```
public void cut(Clipboard clipboard)
```

Cut selected items from the ContentInfoBean and post them into Clipboard.

Specified by:

cut in interface ClipboardUser

Overrides:

cut in class DataItemSinkUIBean

Parameters:

clipboard - the Clipboard object that gets posted to. The actual items posted are contained in a LeifTransferable.

---

### cut

```
public void cut(Clipboard clipboard,  
               Object[] items)
```

Cut the given items from the ContentInfoBean and post them into the given Clipboard only if they occur in the ContentInfoBean.

Specified by:

cut in interface ClipboardUser

Overrides:

cut in class DataItemSinkUIBean

Parameters:

clipboard - the Clipboard object that gets posted to. The actual items posted are contained in a LeifTransferable.  
 items - the array of Object items to cut and posted.

---

### paste

```
public void paste(Clipboard clipboard)
```

Paste the data Objects from the given clipboard. This retrieves the clipboard contents and does a simple add.

Specified by:

paste in interface ClipboardUser

Overrides:

paste in class DataItemSinkUIBean

Parameters:

clipboard - the Clipboard that contains the objects.

See Also:

addRawDataItems(Object[])

---

### contains

```
public final boolean contains(Object[] items)
```

Return true if this ContentInfoBean contains *all* the objects of the given array.

Parameters:

items - an array of objects to locate in the ContentInfoBean.

Returns:

true if there is only one object and it is contained, false otherwise

### containsComponent

```
public boolean containsComponent(Component component)
```

Check if the given component is contained by this InfoBean

Parameters:

component - the Component to check for

Returns:

true if it is contained, else false

### getLeifDataItemMenu

```
public JMenu getLeifDataItemMenu(LeifDataItem dataItem,  
                                boolean showCutPasteItems)
```

Return the data item menu for a LeifDataItem (usually the selected LeifDataItem). This is used by the menubar to add menuitems from this JMenu to a Data menu if there is exactly one DataItemSelected selected.

Overrides:

getLeifDataItemMenu in class DataItemSinkUIBean

Parameters:

dataItem - the LeifDataItem to get the menu for

showCutPasteItems - true to allow cut and paste items to appear, false to omit them.

Returns:

the JMenu for the given LeifDataItem

### isDragEnabled

```
public boolean isDragEnabled()
```

Return true if the default Drag support is enabled.

Overrides:

isDragEnabled in class DataItemSinkUIBean

Returns:

true if drag is enabled, false if not, default is initialized to true.

### setDragEnabled

```
public void setDragEnabled(boolean enabledrag)
```

Set the status of the default Drag support.

Overrides:

setDragEnabled in class DataItemSinkUIBean

Parameters:



enabledrag - true if default drag support should be used, false if not.

---

### setDragOwnerProxy

```
public void setDragOwnerProxy(DragOwner dragProxy)
```

Set a DragOwner "proxy" for this InfoBean.

Overrides:

setDragOwnerProxy in class DataItemSinkUIBean

Parameters:

dragProxy - a DragOwner implementation.

---

### isDropEnabled

```
public boolean isDropEnabled()
```

Return true if the default Drop support is enabled.

Overrides:

isDropEnabled in class DataItemSinkUIBean

Returns:

true if drag is enabled, false if not, default is initialized to true.

---

### setDropEnabled

```
public void setDropEnabled(boolean enabledrop)
```

Set the status of the default Drop support.

Overrides:

setDropEnabled in class DataItemSinkUIBean

Parameters:

enabledrop - true if default drag support should be used, false if not.

---

### setDropOwnerProxy

```
public void setDropOwnerProxy(DropOwner dropProxy)
```

Set a DropOwner "proxy" for this InfoBean.

Overrides:

setDropOwnerProxy in class DataItemSinkUIBean

Parameters:

dropProxy - a DropOwner implementation.

---

## Package com.xis.leif.im

This package contains classes that provide the APIs for using information management in applications.

See:

Description

Interface Summary	
<u><a href="#">Attribute</a></u>	The <code>Attribute</code> class represents an attribute for a particular data type.
<u><a href="#">AttributeAlias</a></u>	The <code>AttributeAlias</code> indicates an alias from 1..n <code>Attributes</code> to a single <code>Attribute</code> , together with a precision level; the higher the precision, the better the alias.
<u><a href="#">AttributeFactory</a></u>	The <code>AttributeFactory</code> class allows an implementor to return an appropriate <code>Attribute</code> for the given <code>LeifDataItem</code> .
<u><a href="#">AttributeLookup</a></u>	The <code>AttributeLookup</code> interface is used to lookup <code>Attribute</code> objects for a particular data item.
<u><a href="#">DisplayLabel</a></u>	The <code>DisplayLabel</code> interface defines methods that are needed for use with <code>DisplayLabelAttributes</code> .
<u><a href="#">Domain</a></u>	This interface describes the basic fields and methods possessed by all <code>Domains</code> .
<u><a href="#">InfoModel</a></u>	The <code>InfoModel</code> interface is the interface that is used to convert raw data items into <code>LeifDataItems</code> .
<u><a href="#">LeifDataItem</a></u>	The <code>LeifDataItem</code> interface represents a simple data item.
<u><a href="#">LeifDataItemObserver</a></u>	This class is used for observing a <code>LeifDataItem</code> to know when it has finished processing an action.
<u><a href="#">LiteDataItem</a></u>	The <code>LiteDataItem</code> interface represents a data item.
<u><a href="#">PropertyProvider</a></u>	If a <code>PropertyProvider</code> implementation is added to services it can be used to replace the standard behavior when a <code>PropertySheetView</code> is opened from a JAF menu or as a default command.
<u><a href="#">RawDataItemLookup</a></u>	The <code>RawDataItemLookup</code> interface is used to look up a raw data item from a unique id.

Class Summary	
<u><a href="#">AbstractAttribute</a></u>	The <code>AbstractAttribute</code> class represents an <code>Attribute</code> .
<u><a href="#">AttributeAliasPluggableService</a></u>	This register <code>AttributeAliases</code> .
<u><a href="#">AttributeDescriptor</a></u>	The <code>AttributeDescriptor</code> class is used to describe an attribute without providing functionality of how to use the attribute.
<u><a href="#">AttributeDescriptorFactory</a></u>	The <code>AttributeDescriptorFactory</code> class is a singleton class used to create or get <code>AttributeDescriptors</code> .
<u><a href="#">AttributeFactoryInfoModelSubset</a></u>	The <code>AttributeFactoryInfoModelSubset</code> class provides an <code>InfoModel</code> that will add the <code>Attributes</code> specified by the <code>AttributeFactories</code> to all <code>LeifDataItems</code> this <code>InfoModel</code> creates.

Figure 31B

<u>AttributeGetRequest</u>	The <u>AttributeGetRequest</u> class is used to package all the necessary parameters for getting the value an attribute.
<u>AttributeLockRequest</u>	The <u>AttributeLockRequest</u> class bundles attributes needed to gain access to locked <u>LeifDataItems</u> .
<u>Attributes</u>	The <u>Attributes</u> is a container for holding attributes.
<u>AttributeSetRequest</u>	The <u>AttributeSetRequest</u> class is used to package all the necessary parameters for setting an attribute.
<u>BaseDataItem</u>	The <u>BaseDataItem</u> is the first wrapper around raw data items.
<u>BaseInfoModel</u>	//PENDING(RK): Any method marked with "PENDING" in the JavaDoc will //likely be removed before XIS is released in final form.
<u>BaseInfoModelServiceProvider</u>	The <u>BaseInfoModelServiceProvider</u> class will provide all the services available from the <u>BaseInfoModel</u> to the given <u>BeanContextServices</u> object.
<u>CollectionProperties</u>	This class populates <u>JAFMenus</u> for generic collections.
<u>DataItemActionManager</u>	The <u>DataItemActionManager</u> class provides some useful static methods for dealing with actions on <u>LeifDataItems</u> .
<u>DataItemActionManager.LeifReferenceActionListener</u>	This class is an <u>actionListener</u> to be used with <u>LeifReference</u> "Load" menus.
<u>DataItemMenuSet</u>	The <u>DataItemMenuSet</u> class is used by the <u>LeifJAFUtilities</u> class to return essentially a <u>DataItem</u> popup menu with annotation.
<u>DataItemMenuSet.Entry</u>	The <u>DataItemMenuSet.Entry</u> class encapsulates a <u>DataItem</u> and it's menu, and also provides some convenience methods for adding additional menu items.
<u>DefaultWrapperAttribute</u>	The <u>DefaultWrapperAttribute</u> class is a generic attribute that is the superclass of all defaults in generated domain attributes.
<u>DisplayLabelAttribute</u>	The <u>DisplayLabelAttribute</u> class is used to display one or more <u>Attribute</u> values in conjunction with string literals specified by users.
<u>DisplayLabelData</u>	The <u>DisplayLabelData</u> class is used by the <u>DisplayLabelAttribute</u> to store a mapping of <u>LeifDataItems</u> to <u>DisplayLabelTemplates</u> .
<u>DisplayLabelTemplate</u>	The <u>DisplayLabelTemplate</u> class is used by the <u>DisplayLabelAttribute</u> to compute and store editing and rendering values for every <u>LeifDataItem</u> that has the attribute.
<u>DomainMethod</u>	Abstractly represents a <u>Domain Method</u> .
<u>DomainMethodDescriptor</u>	The <u>DomainMethodDescriptor</u> class is used to describe a <u>DomainMethod</u> .
<u>DomainMethodDescriptorFactory</u>	The <u>DomainMethodDescriptorFactory</u> class is a singleton class used to create or get <u>DomainMethodDescriptors</u> .
<u>DomainWrapper</u>	This class adds methods to <u>LeifDataItem</u> delegator that are useful in the domain wrappers.
<u>DynamicAttributes</u>	The <u>DynamicAttributes</u> class is used for storing dynamic attributes.

10039300 10001

<u>FieldMetaData</u>	FieldMetaData specifies sorting and subset criteria for an attribute.
<u>FieldMetaDatas</u>	The FieldMetaDatas class represents a collection of FieldMetaData for a data item.
<u>InfoModelDataItem</u>	The InfoModelDataItem allows views to wrap LeifDataItems and add/remove/modify attributes that will only affect that view.
<u>InfoModelSubset</u>	Typically when creating an InfoModel to nest within an existing InfoModel, which is done by ViewUIBeans, an InfoModelSubset is used.
<u>InvalidWrapperAttribute</u>	The InvalidWrapperAttribute class
<u>LeifDataItemComparator</u>	The LeifDataItemComparator compares LeifDataItems by AttributeDescriptor supplied by the user.
<u>LeifDataItemDelegator</u>	Implements the methods in LeifDataItem in a wrapper so you don't have to.
<u>LeifDataItemSorter</u>	The LeifDataItemSorter provides a default sorting tool for all LEIF LeifDataItem objects.
<u>LeifDataItemUpdate</u>	This classes is used with the LeifDataItemObserver.
<u>LeifInitialization</u>	The LeifInitialization class handles some standard initialization for most XIS Applications.
<u>LeifJAFUtilities</u>	The LeifJAFUtilities class provides some useful static methods for LEIF-related JavaBeans Activation Framework (JAF) processing.
<u>LeifJAFUtilities,LeifReferenceActionListener</u>	This class is an ActionListener to be used with LeifReference "Load" menus.
<u>LeifRequest</u>	The LeifRequest class is used to package all the necessary parameters for requesting information for a LeifDataItem.
<u>LeifTransaction</u>	The LeifTransaction class is used to construct a transaction.
<u>LockedLeifDataItem</u>	The LockedLeifDataItem class is used to enable locking on the data item.
<u>MethodRequest</u>	The MethodRequest class is used to package all the necessary parameters for invoking a DomainMethod for a LeifDataItem.
<u>MutableAttributeDescriptor</u>	Mutable subclass of AttributeDescriptor.
<u>ObserverSupport</u>	This class provides useful support for using the LeifDataItemObserver.
<u>RequestPool</u>	The RequestPool class is used to assist Object pooling.
<u>Resources</u>	The Resources class is automatically generated and must be public, but it is intended to be used only by Java's internationalization support classes.
<u>SelectableDataItem</u>	Creates a wrapper around a LeifDataItem for a SelectableInfoModel.
<u>SelectableInfoModel</u>	Manages selections for the selectable leif data items that are contained within this model.
<u>Translator</u>	A major design goal for XIS was to provide the ability to integrate existing data item classes without modifying them.



com.xis.leif.im

## Interface InfoModel

All Superinterfaces:

BeanContextChildOwner, BeanContextChildOwnerDelegator, BeanContextProxy

All Known Implementing Classes:

InfoModelSubset

```
public interface InfoModel
extends BeanContextChildOwnerDelegator
```

The `InfoModel` interface is the interface that is used to convert raw data items into `LeifDataItems`. The `InfoModel` should hold each of these `LeifDataItems` created using weak references so that the data items can be cleaned up when they are no longer being used. //PENDING(RK): Any method marked with "PENDING" in the JavaDoc will likely be removed before LEIF is released in final form.

Since:

LEIF 4.0

Version:

\$Revision: 1.20 \$, \$Date: 2001/08/17 00:54:54 \$

Author:

David Almilli

### Method Summary

void	<code>activateOneOfNService(Object service)</code> //PENDING(RK): This method will probably be removed from <code>InfoModel</code> ! Notify the <code>InfoModel</code> that the given service is the preferred service of its type, and that this particular object should be returned if its class is requested, until removed or until another object of the same type is passed to a future call to this method.
void	<code>addInfoModelListener(InfoModelListener listener)</code> Adds a listener to this <code>InfoModel</code> so that the listener will be informed of changes to the <code>InfoModel</code> .
void	<code>addOneOfNService(Object service)</code> //PENDING(RK): This method will probably be removed from <code>InfoModel</code> ! Add an object as a service to be retrieved by a call to <code>getService()</code> (via <code>BeanContext</code> APIs) on any class that this object implements or extends.
void	<code>clearSelection()</code> Clears the selection.
<code>LeifDataItem</code> (1)	<code>dump()</code> Gives a list of all the <code>LeifDataItems</code> currently in the <code>InfoModel</code> .
<code>EzContext</code>	<code>getEzContext()</code> Gets an <code>EZ Context</code> that corresponds to this <code>InfoModel</code> so the developer can use the <code>EZ APIs</code> .

<u>LeifDataItem</u>	<u>getLeifDataItem(long uid)</u> This will attempt to lookup a LeifDataItem from an id.
<u>LeifDataItem</u>	<u>getLeifDataItem(Object rawDataItem)</u> This will wrap a raw java Object with a LeifDataItem wrapper so you can use it in leif as a data item.
<u>LeifDataItem</u>	<u>getLeifDataItem(Object rawDataItem, boolean create)</u> This will wrap a raw java Object with a LeifDataItem wrapper so you can use it in leif as a data item.
<u>LeifDataItem</u> ( )	<u>getLeifDataItems(Object[] rawDataItems)</u> This convenience method will wrap an array of raw java Objects with LeifDataItem wrappers so you can use them in leif as LeifDataItems.
<u>InfoModel</u>	<u>getParentInfoModel()</u> Provides access to the parent InfoModel that this InfoModel delegates to.
<u>Object[]</u>	<u>getSelectedRawDataItems()</u> Gets the list of all the currently selected items for this InfoModel
<u>Object</u>	<u>getSingleSelectedItem()</u> Get the selected raw data item, if only one.
<u>ViewHost</u>	<u>getViewHost()</u> Gets the ViewHost that this InfoModel is associated with.
<u>void</u>	<u>removeInfoModelListener(InfoModelListener listener)</u> Removes a listener from this InfoModel so that the listener will no longer be informed of changes to the InfoModel.
<u>void</u>	<u>removeOneOfNService(Object service)</u> //PENDING(RK): This method will probably be removed from InfoModel! Remove an object that was a service to be retrieved by a call to getService() (via BeanContext APIs) on any class that this object implements or extends.

Methods inherited from interface com.xis.beans.beancontext.BeanContextChildOwnerDelegator

initializeBeanContextResources, releaseBeanContextResources

Methods inherited from interface com.xis.beans.beancontext.BeanContextChildOwner

getOwnedBeanContextChild

Methods inherited from interface java.beans.beancontext.BeanContextProxy

getBeanContextProxy

## Method Detail

### getLeifDataItem

```
public LeifDataItem getLeifDataItem(long uid)
    throws DataItemNotFoundException
```

This will attempt to lookup a LeifDataItem from an id. If the UID is invalid or there isn't a LeifDataItem that already exists with that given UID, an exception will be thrown.

Parameters:

uid - the unique id for the raw data item.

Returns:

the LeifDataItem with the given UID

---

### getLeifDataItem

```
public LeifDataItem getLeifDataItem(Object rawDataItem)
```

This will wrap a raw java Object with a LeifDataItem wrapper so you can use it in leif as a data item.

**Parameters:**

rawDataItem - the raw data that will be wrapped. *(Note: this should not already be a LeifDataItem)*

**Returns:**

the wrapped data item.

---

### getLeifDataItem

```
public LeifDataItem getLeifDataItem(Object rawDataItem,
                                     boolean create)
```

This will wrap a raw java Object with a LeifDataItem wrapper so you can use it in leif as a data item.

**Parameters:**

rawDataItem - the raw data that will be wrapped. *(Note: this should not already be a LeifDataItem)*  
 create - if false and the LeifDataItem is not already in the model, don't create one and return null

**Returns:**

the wrapped data item, or null if "create" is false and not found

---

### getLeifDataItems

```
public LeifDataItem[] getLeifDataItems(Object[] rawDataItems)
```

This convenience method will wrap an array of raw java Objects with LeifDataItem wrappers so you can use them in leif as LeifDataItems. Note that you can get an array of raw data items often from methods like getMembers(), so this is a useful method to have.

**Parameters:**

rawDataItems - the raw data objects that will be wrapped. *(Note: the objects should not already be LeifDataItems)*

**Returns:**

the corresponding wrapped data item array.

---

### getEzContext

```
public EzContext getEzContext()
```

Gets an EZ Context that corresponds to this InfoModel so the developer can use the EZ APIs.

**Returns:**

the ez context for this info model

---

### getSingleSelectedItem

```
public Object getSingleSelectedItem()
```



Get the selected raw data item, if only one. Else return null.

Returns:

the selected item if there is only one.

### getParentInfoModel

```
public InfoModel getParentInfoModel()
```

Provides access to the parent InfoModel that this InfoModel delegates to. If there is no parent model then this will return null.

Returns:

the parent InfoModel

### clearSelection

```
public void clearSelection()
```

Clears the selection.

### getSelectedRawDataItems

```
public Object[] getSelectedRawDataItems()
```

Gets the list of all the currently selected items for this InfoModel

Returns:

all of the selected data items (as raw data items)

### activateOneOfNService

```
public void activateOneOfNService(Object service)
```

//PENDING(RK): This method will probably be removed from InfoModel! Notify the InfoModel that the given service is the preferred service of its type, and that this particular object should be returned if its class is requested, until removed or until another object of the same type is passed to a future call to this method.

Parameters:

service - the object to become the preferred service

### addOneOfNService

```
public void addOneOfNService(Object service)
```

//PENDING(RK): This method will probably be removed from InfoModel! Add an object as a service to be retrieved by a call to getService() (via BeanContext APIs) on any class that this object implements or extends.

Parameters:

service - the object to be returned when requested

**removeOneOfNService**

```
public void removeOneOfNService(Object service)
```

//PENDING(RK): This method will probably be removed from InfoModel! Remove an object that was a service to be retrieved by a call to getService() (via BeanContext APIs) on any class that this object implements or extends.

**Parameters:**

service - the object to be removed from service

---

**getViewHost**

```
public ViewHost getViewHost()
```

Gets the ViewHost that this InfoModel is associated with. If this InfoModel is not associated with a ViewHost then this will return null.

**Returns:**

the view host that is maintaining this InfoModel.

---

**addInfoModelListener**

```
public void addInfoModelListener(InfoModelListener listener)
```

Adds a listener to this InfoModel so that the listener will be informed of changes to the InfoModel.

**Parameters:**

listener - the listener to add

---

**removeInfoModelListener**

```
public void removeInfoModelListener(InfoModelListener listener)
```

Removes a listener from this InfoModel so that the listener will no longer be informed of changes to the InfoModel.

**Parameters:**

listener - the listener to remove

---

**dump**

```
public LeifDataItem[] dump()
```

Gives a list of all the LeifDataItems currently in the InfoModel. It is highly recommended to use this method only if you absolutely have no other way of accomplishing the task you need to do. Please keep in mind that if you hold onto the LeifDataItems contained in the array returned or if you hold onto the array itself, the items will not be removed from InfoModel until you release them. If you wish to hold onto them, you should wrap them in WeakReference objects.

*Note: When you use the dump() method in combination with the addInfoModelListener so that you can keep track of the same set of LeifDataItems as the InfoModel, you can synchronize on the InfoModel to get the dump and then add a listener to receive events of future changes.*

Example:

```
synchronized(infoModel) {  
    LeifDataItem[] dataItems = infoModel.dump();  
    infoModel.addInfoModelListener(this);  
    for (int i=0; i < dataItems.length; i++) {  
        processItem(dataItems[i]);  
    }  
}
```

Returns:

the list of LeifDataItems currently in the InfoModel.

See Also:

[WeakReference](#)

Figure 1 consists of 12 bar charts, one for each month of the year (January to December). Each chart displays the percentage of total catch for various fish species in the Chesapeake Bay. The species included are Atlantic croaker, Striped bass, Blue crabs, Spot, Weakfish, Rockfish, Atlantic herring, Atlantic menhaden, Atlantic silverside, Atlantic tomcod, Atlantic silverside, and Atlantic tomcod. The charts show seasonal variations in catch percentages for each species.

This package contains classes for handling events in XIS.

### Description

<b><i>InfoModelListener</i></b>	The InfoModelListener is used to monitor changes to an InfoModel.
<b><i>LeifDataItemListener</i></b>	This class is used for listening to LeifDataItems for various events.

<b><u>AttributeChangedEvent</u></b>	An "AttributeChanged" event gets delivered whenever a data item changes an attribute value.
<b><u>ContainerAddedEvent</u></b>	A "ContainerAdded" event gets delivered whenever a data item is contained as a member in a new object.
<b><u>ContainerRemovedEvent</u></b>	A "ContainerRemoved" event gets delivered whenever a data item has been removed as a member from a containing object.
<b><u>DataItemReplacedEvent</u></b>	The DataItemReplacedEvent class is used to indicate member changes of a containing data item.
<b><u>InfoModelEvent</u></b>	The InfoModelEvent gets delivered whenever a LeifDataItem is created by the InfoModel, or when a LeifDataItem has been "lost" by the InfoModel.
<b><u>InfoModelEventSupport</u></b>	The InfoModelEventSupport support class provides basic support for managing listeners on an InfoModel.
<b><u>LeifDataItemAdapter</u></b>	The LeifDataItemAdapter class provides support for setting up a LeifDataItemListener on a data item.
<b><u>LeifEventSupport</u></b>	This is a utility class for XIS developers to use when they want to fire event changes.
<b><u>MemberAddedEvent</u></b>	The MemberAddedEvent class indicates that members were added to this data item.
<b><u>MemberEvent</u></b>	The MemberEvent class is used to indicate members changes of a containing data item.
<b><u>MemberRemovedEvent</u></b>	The MemberRemovedEvent class indicates that the members are being removed from the containing data item.
<b><u>ReferenceAddedEvent</u></b>	The ReferenceAddedEvent class indicates that LeifReferences were added to the LeifDataItem.
<b><u>ReferenceEvent</u></b>	The ReferenceEvent class indicates changes to the LeifReferences of the data item.

Figure 36 B

<u>ReferenceRemovedEvent</u>	The ReferenceRemovedEvent class indicates that LeifReferences were removed from the LeifDataItem.
<u>ReferrerAddedEvent</u>	The ReferrerAddedEvent class indicates that a Referrer was added to the data item.
<u>ReferrerRemovedEvent</u>	The ReferrerRemovedEvent class indicates that a Referrer was removed from the data item.

## Package com.xis.leif.event Description

This package contains classes for handling events in XIS.

10039306-102201  
"FOOTNOT" 90265001

10039306 102201

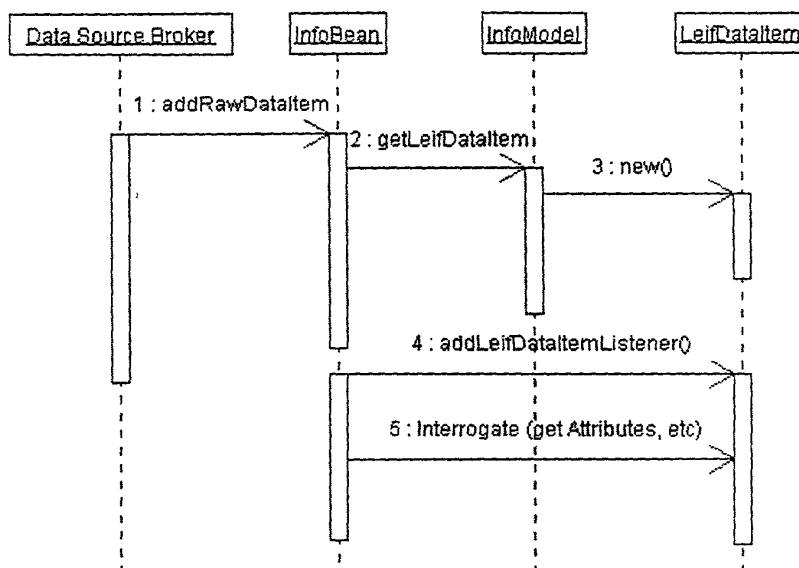


Figure 36C

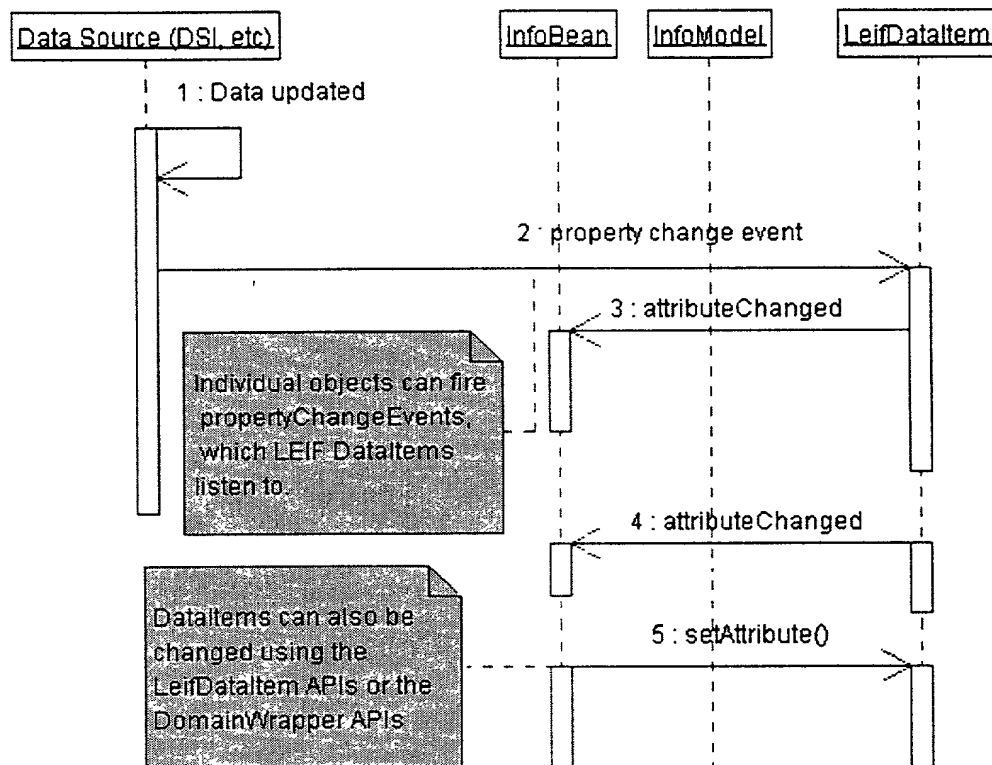


Figure 36D

# Figure 37A

## TestHarness.java

```
/* XIS Tutorial standalone sequence example 5 XIS interfacing. */

import javax.swing.JFrame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
/*{*/
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.awt.Dimension;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import javax.swing.JSplitPane;
import javax.swing.JComponent;
/*}*/
import jclass.chart.JCChart;
import com.xis.leif.im.BaseInfoModel;
import com.xis.plot.PlotInfoBean;
import com.xis.plot.chartviews.LeifChartView;
/*{*/
import com.xis.table.TableInfoBean;
import com.xis.tree.TreeInfoBean;
/*}*/

public class TestHarness {

    public static void main(String[] args) {

        // the plugin manager is only required for more complex applications
        // involving multiple components integrated at runtime
        BaseInfoModel.setStartingPluginManager(false);

/*{*/
        HelloWorld hello1 = new HelloWorld("First HelloWorld object.");
        HelloWorld hello2 = new HelloWorld("Second HelloWorld object.");
        HelloWorld hello3 = new HelloWorld("Third HelloWorld object.");
        HelloWorld hello4 = new HelloWorld("Fourth HelloWorld object.");
        HelloWorld hello5 = new HelloWorld("Fifth HelloWorld object.");
    }
}
```



## Figure 37B

### Continuation of TestHarness.java

```
Object[] helloArray = new Object[] { hello1, hello2, hello3, hello4,
                                     hello5 };
```

```
// create table and plot infobeans to display HelloWorld objects
TableInfoBean table = new TableInfoBean();
```

```
/*}*/
```

```
PlotInfoBean plot = new PlotInfoBean();
plot.setChartType(JCChart.BAR);
// the alternatives are SCATTER_PLOT, PLOT, AREA, PIE, CANDLE,
// and STACKING_BAR, though not all will make sense in this example
```

```
// We can set the attribute for initial display on the plot;
// see step 3 for further comments.
```

```
plot.setYAxisAttribute(
    "com.xis.domains.movement.MovementDomain.speed");
plot.setDynamicAdjustment(true); // so axes track value magnitude
plot.setBarChartAdjusting(true); // needed in some cases for bar chart
```

```
/*{*/
```

```
// a top-level frame as before to hold both plot and table side-by-side
JFrame tablePlotFrame = new JFrame("HelloWorld(s) Table/Plot");
```

```
// use a repaired JSplitPane (see below) to manage the two beans
```

```
SaneJSplitPane splitpane = new SaneJSplitPane(table, plot,
    new Dimension(table.getPreferredSize().width +
        plot.getPreferredSize().width,
        Math.min(table.getPreferredSize().height,
            plot.getPreferredSize().width)),
    0.50);
```

```
tablePlotFrame.getContentPane().add(splitpane);
tablePlotFrame.pack();
```

```
tablePlotFrame.setVisible(true);
```

```
// a tree infobean to display our HelloWorld objects
```

```
TreeInfoBean tree = new TreeInfoBean("HelloWorld(s) Tree");
tree.addRawDataItems(helloArray);
```

```
// a top-level frame to hold our tree infobean
```

```
JFrame treeFrame = new JFrame("HelloWorld(s) Tree");
```

10039306-102201

## Figure 37C

### Continuation of TestHarness.java

```
// avoid placing the windows on top of one another if we can
int cutoffHeight = 424;
if (Toolkit.getDefaultToolkit().getScreenSize().getHeight() >
    cutoffHeight + 200) {
    treeFrame.setLocation(348,cutoffHeight+7);
}
/*}*/

// add a listener for window closing
treeFrame.addWindowListener(
    new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    }
);

// stick the tree infobean in the frame and display it
treeFrame.getContentPane().add(tree);
treeFrame.pack();
treeFrame.setVisible(true);

} // main

/*{*/
/**
 * This class overrides the default JSplitPane to provide a reasonable
 * resize behavior: maintain the left and right panels in the same
 * proportions.
 */
public static final class SaneJSplitPane extends JSplitPane {

    private int    lastWidth;
    private double lastDividerProp;

    public SaneJSplitPane(JComponent leftComponent,
                          JComponent rightComponent,
                          Dimension dims, double startProportion) {
```

## Figure 37D

### Continuation of TestHarness.java

```
super(JSplitPane.HORIZONTAL_SPLIT,
    leftComponent, rightComponent);
setSize(dims);

// Since the JSplitPane doesn't set the lastDividerLocation
// variable, nor does it provide any other easier way to maintain
// the split proportion on resize, we must track the divider
// location ourself.
lastWidth = dims.width;
lastDividerProp = startProportion;
setDividerLocation(startProportion);

// this listens for resize events on the splitpane and makes sure
// we keep same split proportions
addComponentListener(new ComponentAdapter() {
    public void componentResized(ComponentEvent event) {
        setDividerLocation(lastDividerProp);
        lastWidth = (int)event.getComponent().
            getSize().getWidth();
    }
});

// only way to know if divider moved by user is to listen for
// resize events on the components; this isn't foolproof (since
// resizes can come from other sources) but it works well enough
leftComponent.addComponentListener(new ComponentAdapter() {
    public void componentResized(ComponentEvent event) {
        // we add in getDividerSize() / 4 to compensate for a
        // bug in JSplitPane which doesn't take account of the
        // divider width in location-proportion conversions
        lastDividerProp = (double)(getDividerLocation() +
            (getDividerSize() / 4)) / lastWidth;
    }
});

}

}
/*}*/
}
```

Figure 38A

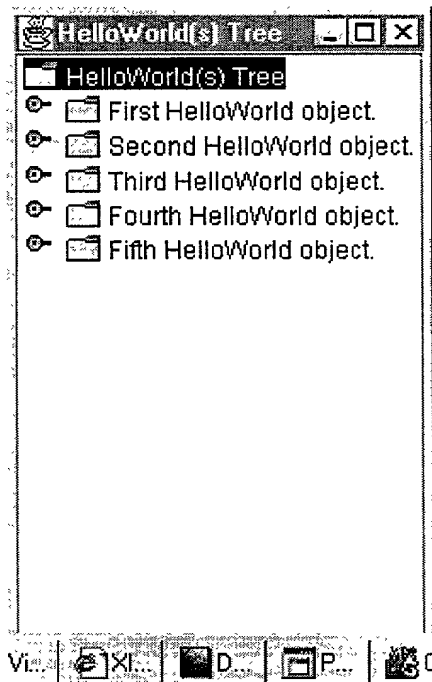
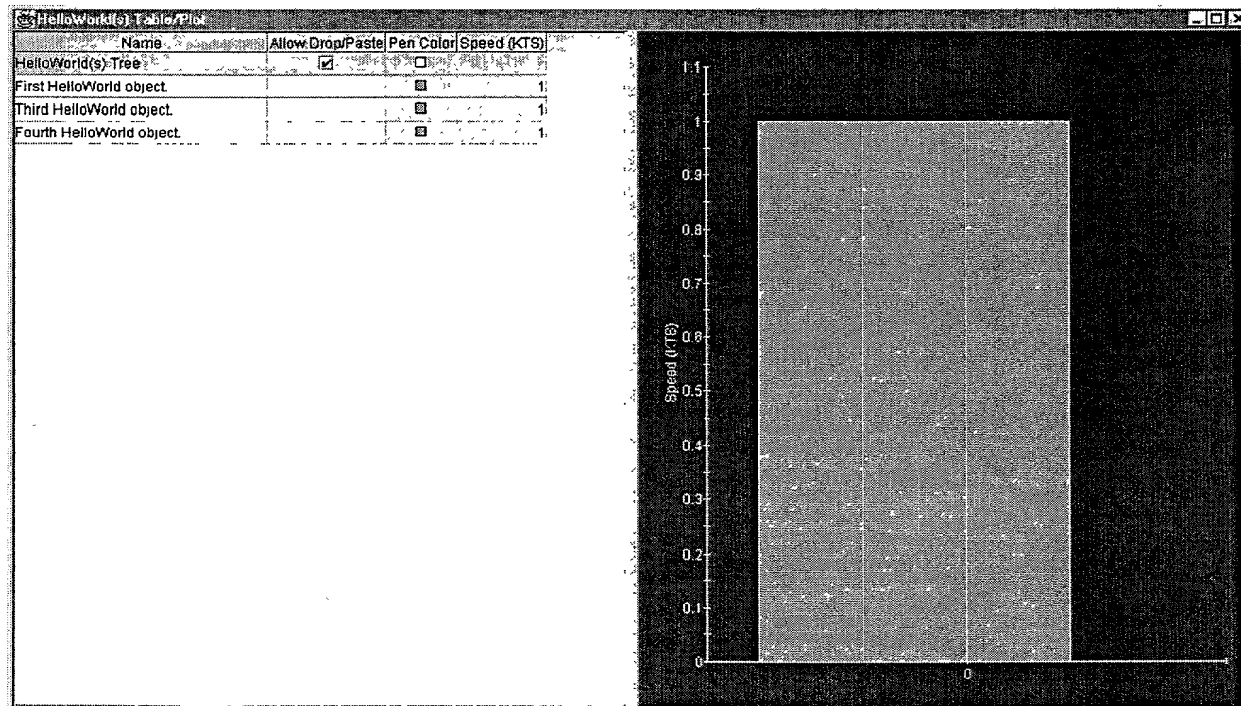


Figure 38B



10039306-10201

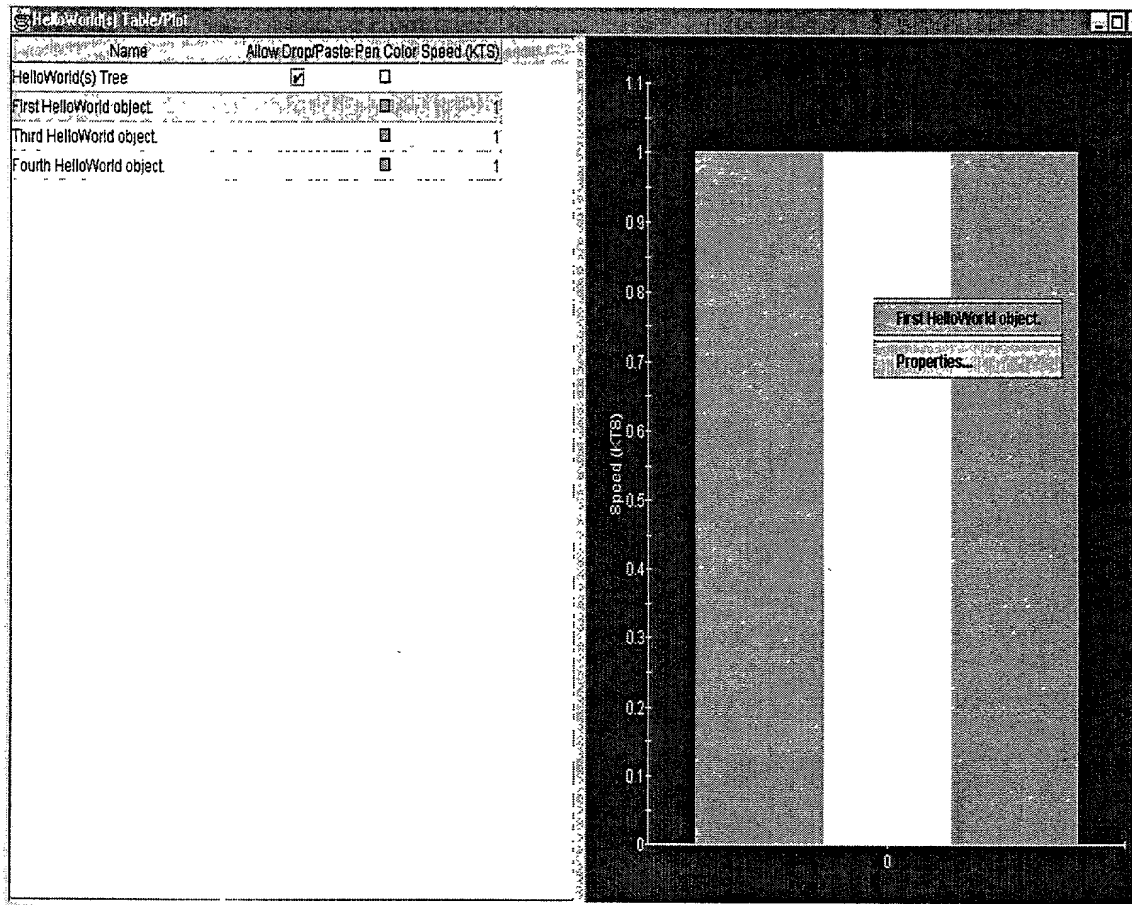


Figure 38C

10039306-102201


Property	Value
Name	First HelloWorld object.
Speed (KTS)	1
Pen Color	

Figure 38D

Fig. 39A

com.xis.leif.im

## Interface AttributeAlias

public interface AttributeAlias

The AttributeAlias indicates an alias from 1..n Attributes to a single Attribute, together with a precision level; the higher the precision, the better the alias. The alias allows a caller to query for one of the 1..n "from" attributes and get the value stored by the data item under the "to" attribute, possibly mediated by some conversion, such as a units transformation.

If the converted or calculated value cannot be determined, then the Attribute#getValue() method should throw an UnconvertibleAliasException. This Exception is a subclass of the UndefinedLeifAttributeException which is typically thrown by normal Attributes in this case, and it can provide a descriptive message indicating the source of the incompatibility.

The utility of attribute aliases may be seen by considering the following example:

The user has performed a query from an external data source and retrieved a set of Airfields, indexed by an ICAO identifier. The user now wants to get the list of Aircraft at one of the airfields. There is a local aircraft database with a foreign key field for Airfields, but that key is a WAC identifier, not ICAO.

Assumption: The application was NOT written ahead of time to know about these two databases or their ID types. Instead, what you have is an XIS "LeifDataItem" for the Airfield, and you have an XIS InfoBean for the Aircraft query form.

What you want to do is to copy (or "drop") the Airfield data item into the "WAC" field in the query form. In doing this, the Form will ask the data item for its "WAC" attribute (because this is all it knows about). It uses the "getWAC()" method from some domain (say, the AirfieldDomain).

The way this could work is that there would have to be an AttributeAlias defined to convert ICAO to WAC - or, more specifically, AviationDomain. ICAO to AirfieldDomain. WAC. The AttributeAlias returns an Attribute object that knows how to transform ICAOs to WACs (e.g., by accessing a conversion table). The Attribute, in turn, has a getValue() method to execute that transformation and return the WAC.

This process would be entirely transparent to the user, or even the caller, who would just see a result returned from the getWAC() method. In cases where the conversion was not possible, the UnconvertibleAliasException would be thrown, possibly providing informative information to the caller or user.

Finally, note that due to the way the mechanism is set up (using resources and a PluggableService), this AttributeAlias can be installed as a separate module without requiring any re-coding or re-compilation of the existing application.

## Method Summary

<u>AttributeDescriptor</u> []	<u>getAliasedFrom</u> () This indicates which AttributeDescriptors (which in turns means which Attributes) are required for the alias.
<u>AttributeDescriptor</u>	<u>getAliasedTo</u> () This indicates the AttributeDescriptor that this AttributeAlias is for.
<u>Attribute</u>	<u>getAttribute</u> () Get the Attribute object that is the alias Attribute.
<u>int</u>	<u>getPrecisionPriority</u> () This indicates the precision of the AttributeAlias.

## Method Detail

### getPrecisionPriority

```
public int getPrecisionPriority()
```

This indicates the precision of the AttributeAlias. The higher the number the better the alias. This number is used to determine which AttributeAlias to use when there are more than one alias for a given Attribute.

**Returns:**

the precision of the alias.

### getAliasedFrom

```
public AttributeDescriptor[] getAliasedFrom()
```

This indicates which AttributeDescriptors (which in turns means which Attributes) are required for the alias.

**Returns:**

the list of descriptors required for this alias.

### getAliasedTo

```
public AttributeDescriptor getAliasedTo()
```

This indicates the AttributeDescriptor that this AttributeAlias is for.

**Returns:**

the descriptor that this alias is for.

Fig. 39B





FIG. 40

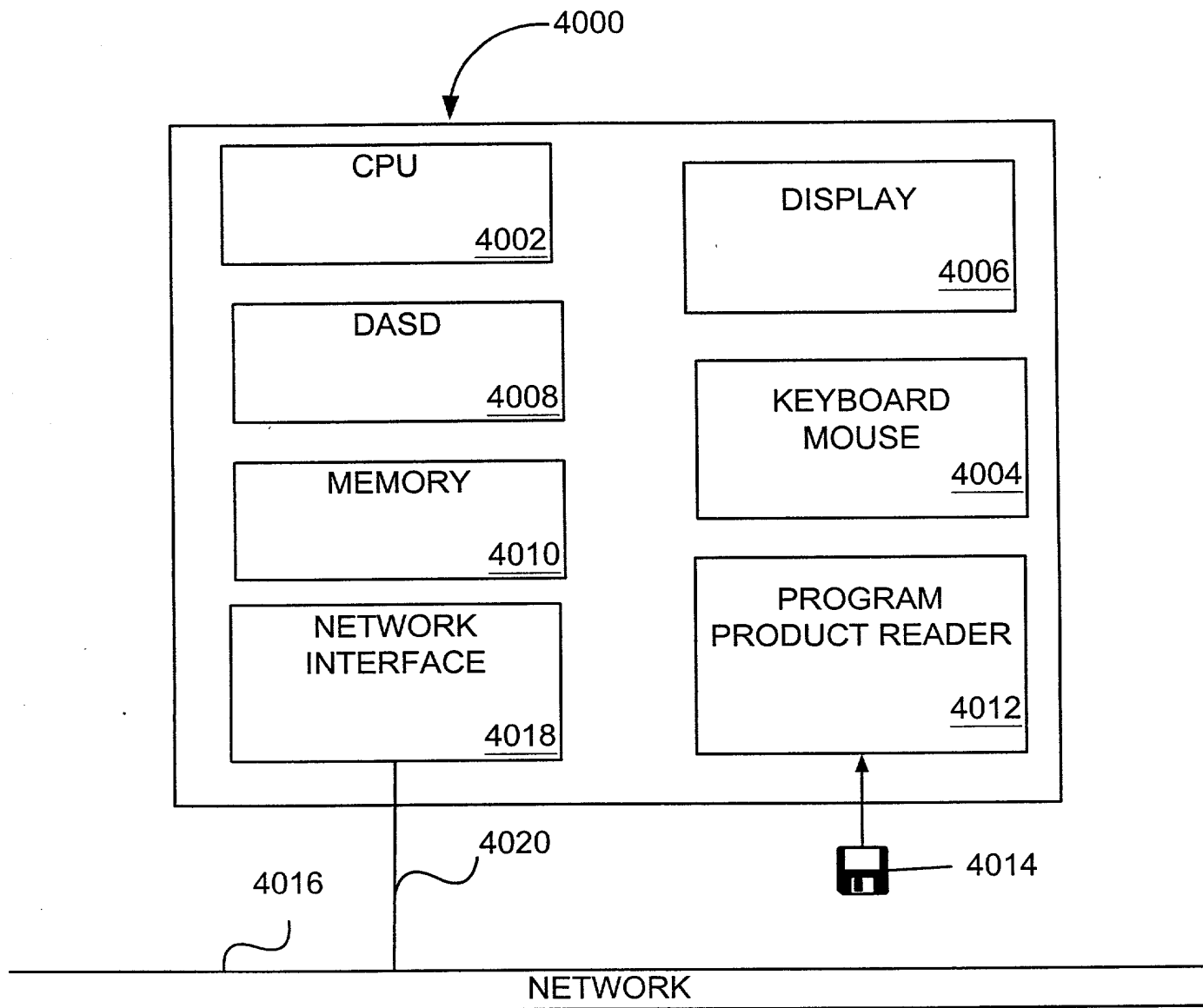


Figure 40